Visualizing the flow of a binary program using tcl

(by Uli Ender, Munich, Germany)

In my talk I will present tcl-scripts that load a binary program and run that program instruction by instruction.

The memory used by the program is visualized in a canvas-widget as a ppm-picture, that is actualized after each programming step.

Output of the program is shown in a text-widget.

A DEMO-program of a 64-bit-processor will be started by the following script:

```
# virtual computer for AMD x86_64

file copy -force demo-program    1    ; # the main-script
file copy -force tests           2    ; # the binary to be visualized
file copy -force Text1           3    ; # the source to be processed by the binary
                                 4    ; # will contain the binary after visualization
file copy -force empty-list      5    ; # a list of parameters (registers number of instructions) before visualization
file copy -force empty-picture   6    ; # the picture of memory before visualization
                                 7    ; # will contain the list of registers and numbers of instructions afterwards
                                 8    ; # will contain the picture of memory afterwards
source 1
```

1 main-script | 2 binary to be vizualised | 3 source text | 4 binary afterwards

```
# virtueller E-Computer
# 64-bit instrukcioj
global p sss stack in zwww

…

init
loop
state
```

```
00000000  7f 45 4c 46 02 01 01 00  00 45 73 70 65 72 6f 20  |.ELF.....Espero |
00000010  02 00 3e 00 01 00 00 00  00 10 40 00 00 00 00 00  |..>.......@.....|
00000020  40 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |@...............|
00000030  00 00 00 00 40 00 38 00  01 00 00 00 00 00 00 00  |....@.8.........|
00000040  01 00 00 00 07 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 40 00 00 00 00 00  00 00 40 00 00 00 00 00  |..@.......@.....|
00000060  00 00 00 00 50 01 00 00  00 00 00 00 50 01 00 00  |....P.......P...|
00000070  00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000200  1a 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000300  00 04 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |..@.............|
00000310  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  0d 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000610  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000700  00 08 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |..@.............|
00000710  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800  68 61 6c 6c 6f 20 4c 65  75 74 65 21 0a 00 00 00  |hallo Leute!....|
00000810  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  b8 00 00 00 00 bf 00 00  00 00 48 8b 14 25 00 02  |..........H..%..|
00001010  40 00 48 8b 34 25 00 03  40 00 0f 05 b8 01 00 00  |
@.H.4%..@.......|
00001020  00 bf 01 00 00 00 48 8b  14 25 00 06 40 00 48 be  |......H..%..@.H.|
00001030  00 07 40 00 00 00 00 00  0f 05 a1 00 07 40 00 00  |..@..........@..|
00001040  00 00 00 a3 00 0c 40 00  00 00 00 00 00 00 00 00  |......@.........|
00001050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00010000
```

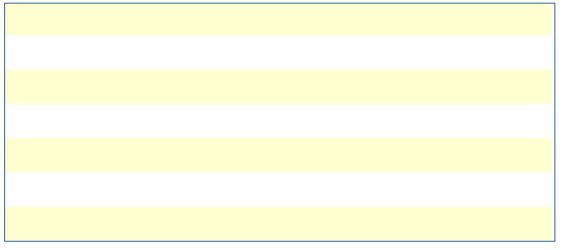
hello world! How are you?

```
00000000  7f 45 4c 46 02 01 01 00  00 45 73 70 65 72 6f 20  |.ELF.....Espero |
00000010  02 00 3e 00 01 00 00 00  00 10 40 00 00 00 00 00  |..>.......@.....|
00000020  40 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |@...............|
00000030  00 00 00 00 40 00 38 00  01 00 00 00 00 00 00 00  |....@.8.........|
00000040  01 00 00 00 07 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 40 00 00 00 00 00  00 00 40 00 00 00 00 00  |..@.......@.....|
00000060  00 00 00 00 50 01 00 00  00 00 00 00 50 01 00 00  |....P.......P...|
00000070  00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000200  1a 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000300  00 04 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |..@.............|
00000310  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  68 65 6c 6c 6f 20 77 6f  72 6c 64 21 20 48 6f 77  |hello world! How|
00000410  20 61 72 65 20 79 6f 75  3f 0a 00 00 00 00 00 00  | are you?.......|
00000420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  0d 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000610  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000700  00 08 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |..@.............|
00000710  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000800  68 61 6c 6c 6f 20 4c 65  75 74 65 21 0a 00 00 00  |hallo Leute!....|
00000810  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000c00  00 08 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |..@.............|
00000c10  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001000  b8 00 00 00 00 bf 00 00  00 00 48 8b 14 25 00 02  |..........H..%..|
00001010  40 00 48 8b 34 25 00 03  40 00 0f 05 b8 01 00 00  |
@.H.4%..@.......|
00001020  00 bf 01 00 00 00 48 8b  14 25 00 06 40 00 48 be  |......H..%..@.H.|
00001030  00 07 40 00 00 00 00 00  0f 05 a1 00 07 40 00 00  |..@..........@..|
00001040  00 00 00 a3 00 0c 40 00  00 00 00 00 00 00 00 00  |......@.........|
00001050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00010000
```

5 empty-list

```
set p(in-pointer)          0

set p(bef)                {}
set p(pc)                  0
set p(rax)                 0
...
set p(rdx)                 0
set p(rdi)                 0
set p(rsi)                 0
…
set p(null)                0
set p(transigo)            0
set p(sp)                  0
set stack                { 0 0 0 .. }
set p(z-gesamt)            0
...
set p(z-syscall)           0
set p(z-lade-a)            0
set p(z-speichere-a)       0
...
set p(z-mov_rax_32bit)     0
...
set p(z-mov_rdx_{32bit})   0
set p(z-mov_rdi_32bit)     0
set p(z-mov_rsi_{32bit})   0
set p(z-mov_rsi_{64bit})   0
...
```

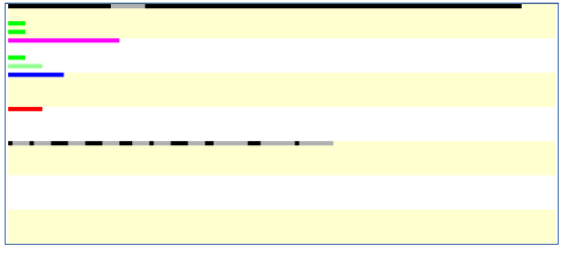
6 empty-picture

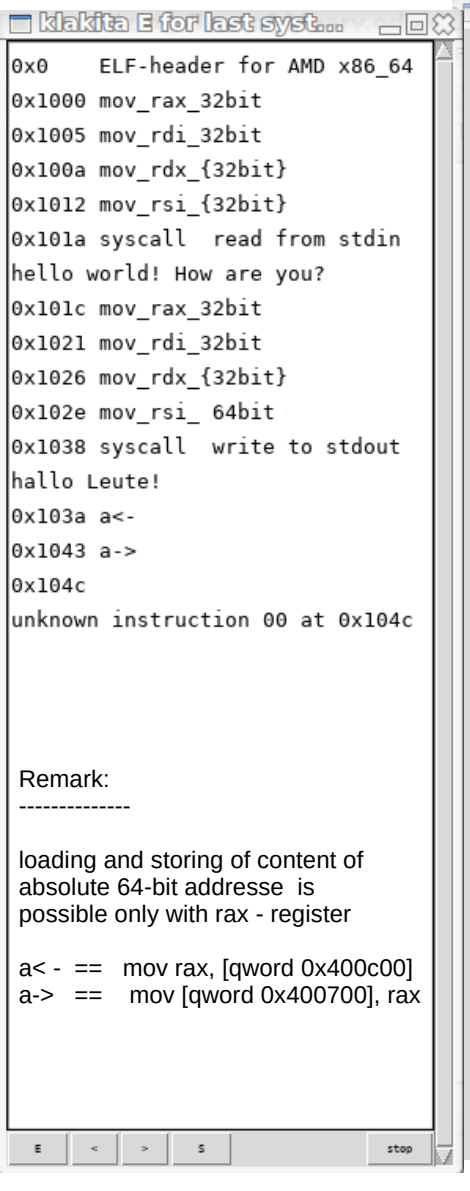


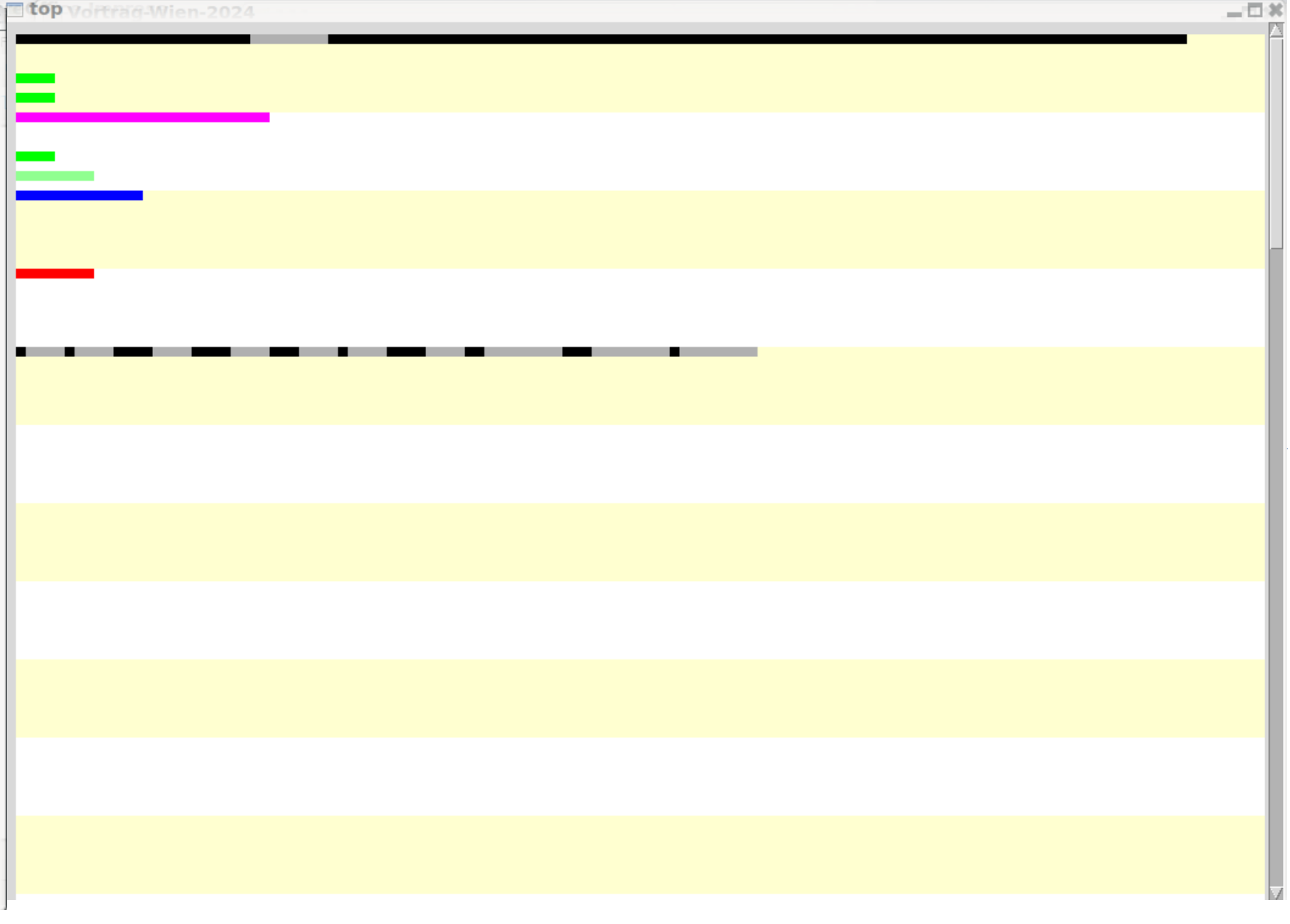
7 state afterwards

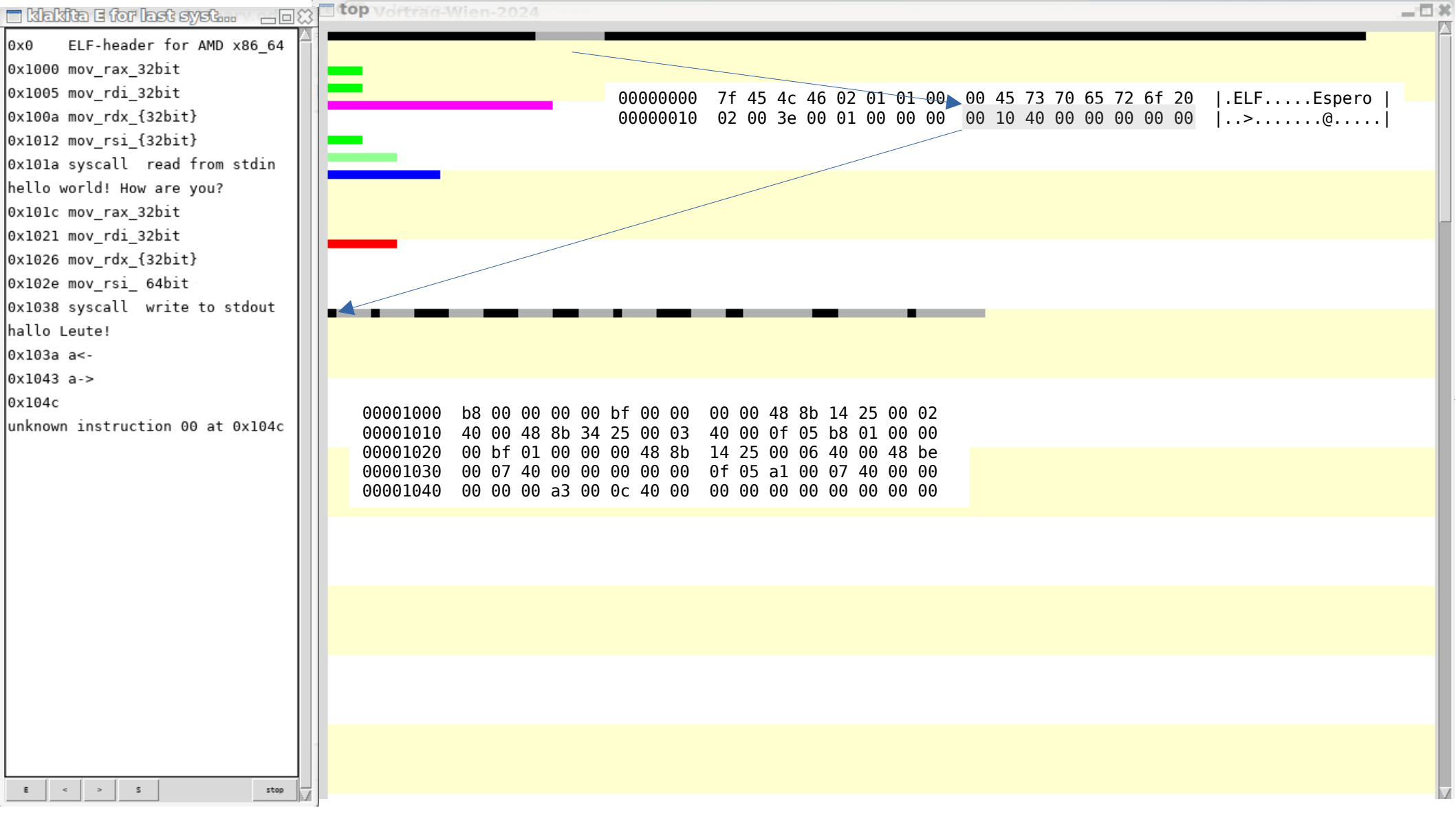
```
set p(in-pointer)          26

set p(bef)                { = A3   a-> }
set p(pc)                  0x40104c
set p(rax)                 0x0000000000400800
...
set p(rdx)                 0x0000000d
set p(rdi)                 0x00000001
set p(rsi)                 0x0000000000400800
…
set p(null)                0
set p(transigo)            0
set p(sp)                  0
set stack                { 0 0 0 ...}
set p(z-gesamt)            12
...
set p(z-syscall)           2
set p(z-lade-a)            1
set p(z-speichere-a)       1
...
set p(z-mov_rax_32bit)     2
...
set p(z-mov_rdx_{32bit})   2
set p(z-mov_rdi_32bit)     2
set p(z-mov_rsi_{32bit})   1
set p(z-mov_rsi_{64bit})   1
...
```

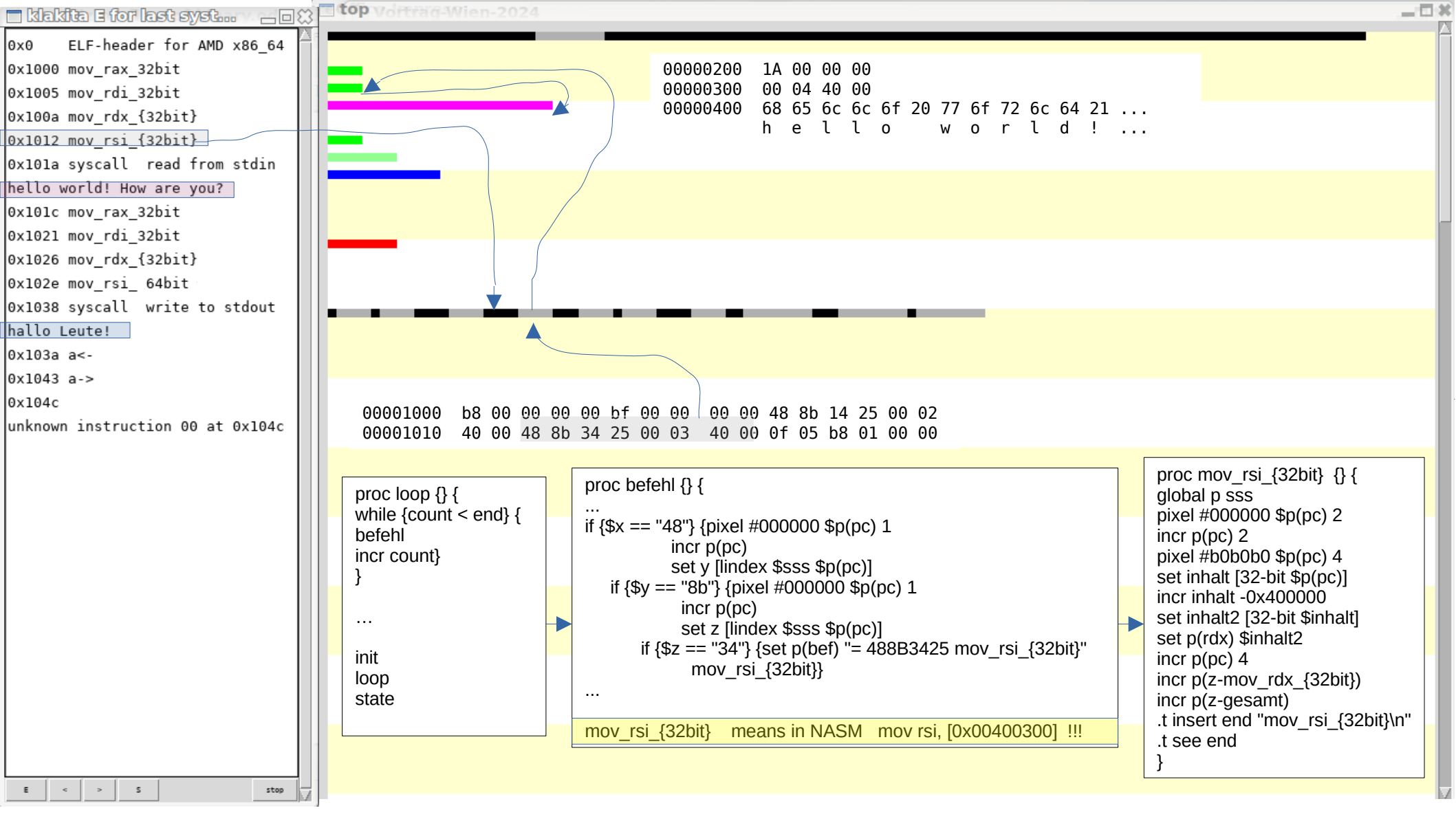
8 picture afterwards




klakita E for last syst...
0x0 ELF-header for AMD x86_64
0x1000 mov_rax_32bit
0x1005 mov_rdi_32bit
0x100a mov_rdx_{32bit}
0x1012 mov_rsi_{32bit}
0x101a syscall read from stdin
hello world! How are you?
0x101c mov_rax_32bit
0x1021 mov_rdi_32bit
0x1026 mov_rdx_{32bit}
0x102e mov_rsi_ 64bit
0x1038 syscall write to stdout
hallo Leute!
0x103a a<-
0x1043 a->
0x104c
unknown instruction 00 at 0x104c
Remark:
---
loading and storing of content of
absolute 64-bit addresse is
possible only with rax - register
a< - == mov rax, [qword 0x400c00]
a-> == mov [qword 0x400700], rax
E
<
>
S
stop


top
Vortrag-Wien-2024


klakita E for last syst...
0x0 ELF-header for AMD x86_64
0x1000 mov_rax_32bit
0x1005 mov_rdi_32bit
0x100a mov_rdx_{32bit}
0x1012 mov_rsi_{32bit}
0x101a syscall read from stdin
hello world! How are you?
0x101c mov_rax_32bit
0x1021 mov_rdi_32bit
0x1026 mov_rdx_{32bit}
0x102e mov_rsi_ 64bit
0x1038 syscall write to stdout
hallo Leute!
0x103a a<-
0x1043 a->
0x104c
unknown instruction 00 at 0x104c
E
<
>
S
stop
top
00000000 7f 45 4c 46 02 01 01 00 00 45 73 70 65 72 6f 20 |.ELF.....Espero |
00000010 02 00 3e 00 01 00 00 00 00 10 40 00 00 00 00 00 |..>.....@.....|
00001000 b8 00 00 00 00 bf 00 00 00 00 48 8b 14 25 00 02
00001010 40 00 48 8b 34 25 00 03 40 00 0f 05 b8 01 00 00
00001020 00 bf 01 00 00 00 48 8b 14 25 00 06 40 00 48 be
00001030 00 07 40 00 00 00 00 00 0f 05 a1 00 07 40 00 00
00001040 00 00 00 a3 00 0c 40 00 00 00 00 00 00 00 00 00


klakita E for last syst...
0x0 ELF-header for AMD x86_64
0x1000 mov_rax_32bit
0x1005 mov_rdi_32bit
0x100a mov_rdx_{32bit}
0x1012 mov_rsi_{32bit}
0x101a syscall read from stdin
hello world! How are you?
0x101c mov_rax_32bit
0x1021 mov_rdi_32bit
0x1026 mov_rdx_{32bit}
0x102e mov_rsi_ 64bit
0x1038 syscall write to stdout
hallo Leute!
0x103a a<-
0x1043 a->
0x104c
unknown instruction 00 at 0x104c
E
<
>
S
stop
top
Vortrag-Wien-2024
00000200 1A 00 00 00
00000300 00 04 40 00
00400 68 65 6c 6c 6f 20 77 6f 72 6c 64 21 ...
h e l l o w o r l d ! ...
00001000 b8 00 00 00 00 bf 00 00 00 00 48 8b 14 25 00 02
00001010 40 00 48 8b 34 25 00 03 40 00 0f 05 b8 01 00 00
proc loop {} {
while {count < end} {
befehl
incr count}
}
…
init
loop
state
proc befehl {} {
...
if {$x == "48"} {pixel #000000 $p(pc) 1
incr p(pc)
set y [lindex $sss $p(pc)]
if {$y == "8b"} {pixel #000000 $p(pc) 1
incr p(pc)
set z [lindex $sss $p(pc)]
if {$z == "34"} {set p(bef) "= 488B3425 mov_rsi_{32bit}"
mov_rsi_{32bit}}
...
mov_rsi_{32bit} means in NASM mov rsi, [0x00400300] !!!
proc mov_rsi_{32bit} {} {
global p sss
pixel #000000 $p(pc) 2
incr p(pc) 2
pixel #b0b0b0 $p(pc) 4
set inhalt [32-bit $p(pc)]
incr inhalt -0x400000
set inhalt2 [32-bit $inhalt]
set p(rdx) $inhalt2
incr p(pc) 4
incr p(z-mov_rdx_{32bit})
incr p(z-gesamt)
.t insert end "mov_rsi_{32bit}\n"
.t see end
}

Scripts to update the picture

initialize the picture

```
global img
set img [image create photo meinphoto -file 6]
destroy .top
toplevel .top
wm geometry .top 1050x720+720+0
canvas .top.c -scrollregion {0 0 1050 2880} -width 1050 -height 720
.top.c create image 10 10 -anchor nw -image $img
meinphoto copy meinphoto -zoom 1
scrollbar .top.s -command ".top.c yview"
pack .top.s -side right -fill y
pack .top.c -expand yes -fill both
.top.c config -yscrollcommand ".top.s set"


# pixel #000000 n z       ;# black   executing
# pixel #b0b0b0 n z       ;# grey          „ „
# pixel #f f 0000 n z     ;# red      writing   to
# pixel #00f f 00 n z     ;# green   reading from
# pixel #0000f f  n z     ;# blue    writing   to     std-out
# pixel #f f 00f f  n z   ;# violet   reading from std-in
# pixel #f f f f 00 n z   ;# yellow  empty
# pixel #f f f f f f  n z ;# white   empty
```

add number of coloured pixels to the picture

```
proc pixel {farbe n z} {
# global img
set m 0
while {$m < $z} {
incr m
set y1 [expr (int($n/128))]
set x1 [expr ($n-$y1*128)]
#  .t insert end "#  x1= $x1 y1= $y1\n"
set y1 [expr ($y1*8)]
set x1 [expr ($x1*8)]
set y2 $y1
incr y2 8
set x2 $x1
incr x2 8
if {$farbe == "#000000" || $farbe == "#00ff00"} {
set f [meinphoto get $x1 $y1]
# .t insert end "# $f\n"
# .t see end
if {$f == "0 0 0"} { set farbe #b0b0b0}
if {$f == "176 176 176"} { set farbe #000000}
if {$f == "0 255 0"} { set farbe #90ff90}
if {$f == "135 255 135"} { set farbe #00ff00}
}
meinphoto put $farbe -to $x1 $y1 $x2 $y2
incr n    }  }
```

Uses of visualization

I used it to optimize a tiny compiler that compiles itself by reading its source text.

The smallest extensible 64-bit compiler I have achieved:

1	instruction == . (point)
1	data-type == byte(s) (number, code, name)
212	lines of code (nasm assembler)
802	bytes (zipped file of 64k)
79398	instructions to compile the compiler
40 µsec	compilation time
1.98	billion instructions/sec
Computer:	HP with Linux Knoppix 9.3 – Celeron N4000 @1,1GHz 2 Cores 2 threads
	Kernel 5.16.12-64 2488 Mhz 8GB ram
	cache size 4096 KB level 1 cache 2x24KB data 2x32KB instr

COMPILING a COMPILER I

Source of the Compiler: Besides the instruction . already known 7 other words are defined

.

. creates a new entry in the library

(

(permits to write comments
comments end with a)

za

start-time

zb

end-time & difference

bf1

bf2

bf3

move end of compiler to new position

Ende

copy new compiler to std-out and exit program

```
.
28 20    4D 8B 18
         49 FF C3
         41 8A 1B
         80 FB 29
         75 F5
         4D 89 18
         C3
.
(
Now it is possible to write
comments within round brackets
)

7A 61 20 B8 60 00 00 00
         48 BF 10 24 40 00 00 00 00 00
         48 BE 20 24 40 00 00 00 00 00
         0F 05
         C3
.
7A 65 20 B8 60 00 00 00
         48 BF 20 24 40 00 00 00 00 00
         48 BE 30 24 40 00 00 00 00 00
         0F 05
         48 8B 04 25 28 24 40 00
         48 2B 04 25 18 24 40 00
         48 89 04 25 08 24 40 00
         C3
.
62 66 31 20  41 BC 00 20 40 00
             C3
.
62 66 32 20  41 BC 00 21 40 00
             C3
.
62 66 33 20  41 BC 00 30 40 00
             C3
.
45 6E 64 65 20  B8 01 00 00 00
                BF 01 00 00 00
                BA 00 14 00 00
                48 BE 00 20 40 00 00 00 00 00
                0F 05
                B8 3C 00 00 00
                BF 00 00 00 00
                0F 05
                C3
.
```

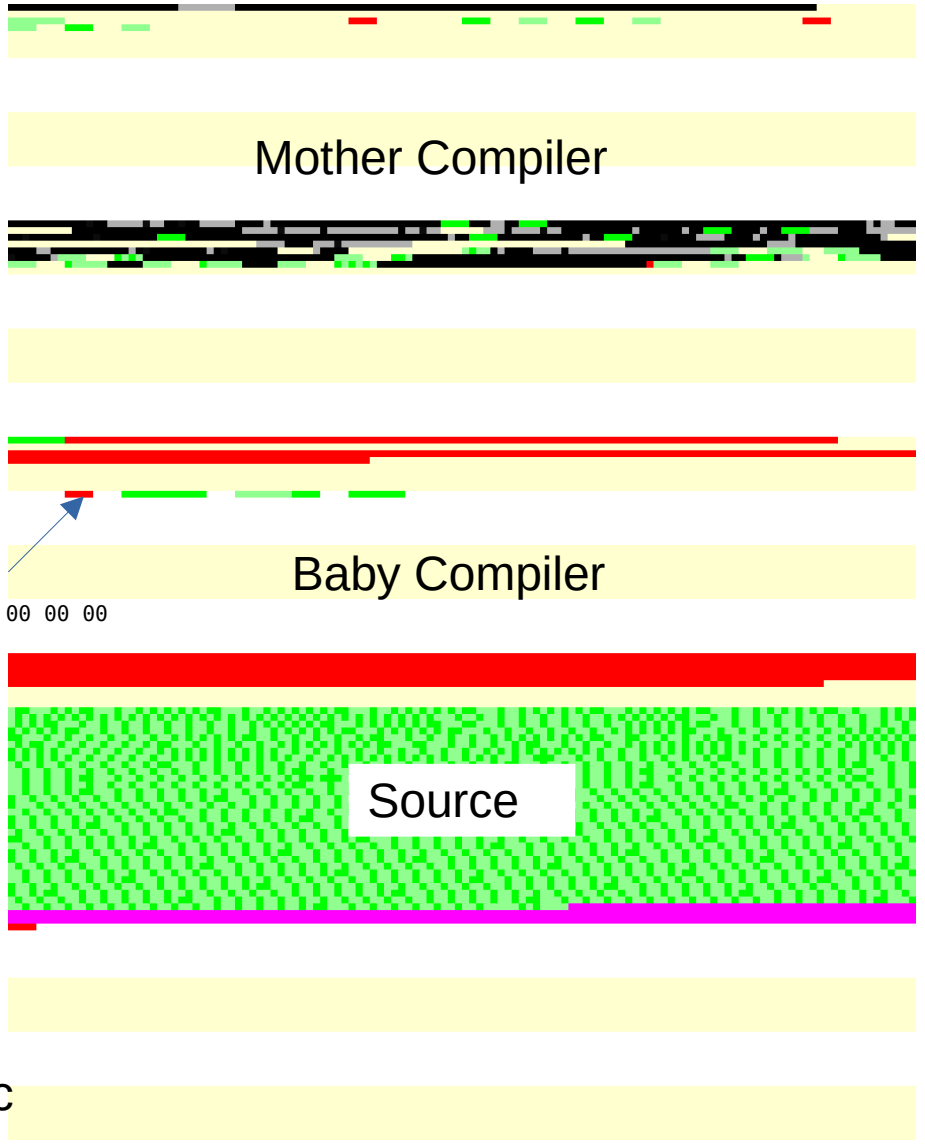
```
za
bf1
7F 45 4C 46 02 01 01 00  00 45 73 70 65 72 6F 20
02 00 3E 00 01 00 00 00  D7 11 40 00 00 00 00 00
40 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 40 00 38 00  01 00 00 00 00 00 00 00
01 00 00 00 07 00 00 00  00 00 00 00 00 00 00 00
00 00 40 00 00 00 00 00  00 00 40 00 00 00 00 00
00 00 00 50 01 00 00 00  00 00 00 50 01 00 00 00
00 10
bf2
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 01 40 00 00 00 00 00
6B 12 40 00 00 00 00 00  00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00 10 00 00 00 00 00 00  63 12 40 00 00 00 00 00
BC 0A 00 00 00 00 00 00  23 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
10 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
01 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00
65 72 61 72 6F 21 0A 00  00 00 00 00 00 00 00 00
```

```
bf3
4D 8B 18 49 FF C3 41 8A  1B 80 FB 21 73 F5 41 8A
1B 80 FB 00 74 0E 80 FB  21 73 05 49 FF C3 EB EE
4D 89 18 C3 48 B8 00 01  40 00 00 00 00 00 4C 29
C0 75 4D 48 8B 04 25 30  01 40 00 48 05 95 2D 00
00 49 89 00 48 89 04 25  48 01 40 00 B8 00 00 00
00 BF 00 00 00 00 48 8B  14 25 50 01 40 00 48 8B
34 25 48 01 40 00 0F 05  48 8B 1C 25 48 01 40 00
48 01 C3 B8 00 00 00 00  48 89 03 4D 8B 18 EB 8E
49 83 E8 08 E9 77 FF FF  FF 4D 8B 18 4C 8B 2C 25
58 01 40 00 4D 89 EE 4D  89 DF 49 FF CF 49 83 C6
08 41 8A 0E 80 F9 21 72  1A 49 FF C7 41 8A 1F 38
D9 74 21 4D 8B 6D 00 49  83 FD 00 75 D7 B8 00 00
00 00 C3 49 FF C7 41 8A  1F 80 FB 21 73 E5 B8 01
00 00 00 C3 49 FF C6 EB  C8 B8 00 00 00 00 48 89
04 25 70 01 40 00 49 8B  00 48 89 04 25 40 01 40
00 48 8B 04 25 40 01 40  00 8A 18 80 EB 30 72 6F
80 FB 0A 72 0D 80 EB 11  72 65 80 FB 18 73 60 80
C3 0A 88 1C 25 88 01 40  00 8A 04 25 80 01 40 00
28 C3 74 4B 48 8B 04 25  70 01 40 00 48 0F AF 04
25 80 01 40 00 48 03 04  25 88 01 40 00 48 89 04
25 70 01 40 00 48 8B 04  25 40 01 40 00 48 FF C0
48 89 04 25 40 01 40 00  8A 18 80 FB 21 72 02 EB
90 4C 8B 14 25 70 01 40  00 B8 01 00 00 00 C3 B8
00 00 00 00 C3 49 FF C6  41 FF D6 C3 B8 01 00 00
00 BF 01 00 00 00 BA FF  FF 00 00 48 BE 00 00 40
00 00 00 00 00 0F 05 B8  3C 00 00 00 BF 00 00 00
00 0F 05 48 8B 04 25 90  01 40 00 48 83 F8 01 75
08 45 88 14 24 49 FF C4  C3 4D 89 14 24 49 83 C4
08 B8 01 00 00 00 48 89  04 25 98 01 40 00 48 89
04 25 90 01 40 00 C3 4C  8B 1C 25 30 01 40 00 4C
8B 24 25 30 01 40 00 4C  89 24 25 00 01 40 00 49
B8 00 01 40 00 00 00 00  00 41 B9 00 00 00 00 E8
02 00 00 00 EB F9 E8 F5  FD FF FF E8 C9 FE FF FF
48 83 F8 00 75 15 E8 6E  FE FF FF 48 83 F8 00 74
05 E9 4F FF FF FF E9 51  FF FF FF E9 73 FF FF FF
00 00 00 00 00 00 00 00  30 12 40 00 00 00 00 00
2E 20 48 8B 04 25 58 01  40 00 49 89 04 24 4C 89
24 25 58 01 40 00 49 83  C4 08 4C 89 24 25 30 01
40 00 C3 38 12 40 00 00  00 00 00 00 00 00 00 00
ze
Ende
```

Compiling a Compiler II

A) in Tcl/Tk
Number of instructions = 98759-19508 = 79398

```
set p(bef)              { syscall  exit program performed }
set p(pc)                 0x40135b
set p(rax)                0x000000003c
set p(rbx)                0x40500a
set p(rcx)                0x20
set p(rdx)                0x0000001400
set p(rdi)                0x0000000000
set p(rsi)                0x402000
set p(rbp)                0
set p(rsp)                0
set p(r8)                 0x400100
set p(r9)                 0x0000000000
set p(r10)                0x0000000000
set p(r11)                0x404ec7
set p(r12)                0x403270
set p(r13)                0x0000401326
set p(r14)                0x401333
set p(r15)                0x404ecb
set p(null)               0
set p(transigo)           0
set p(sp)                 2
set stack               { 0x1204 0x117b 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 }
set p(z-gesamt)           98759
```



B) using the binary itself in console
./Mother-compiler < Source > Baby-compiler
Compilation time = 0x31 = 49-9 = µs

C) ==> 79398/40*1000000 = 1,985 billion instr/sec

Thanks for listening!

Contact: uli.ender@esperanto.de

BACKUP

The following screenshots show the states of compiling
after always 1000000-instruction-steps

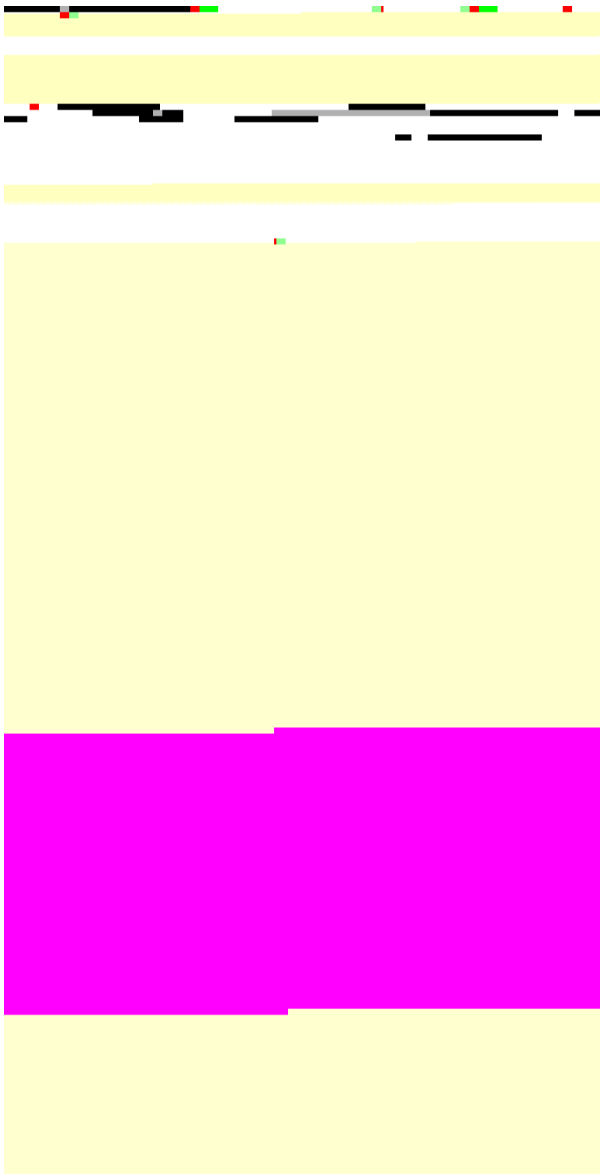
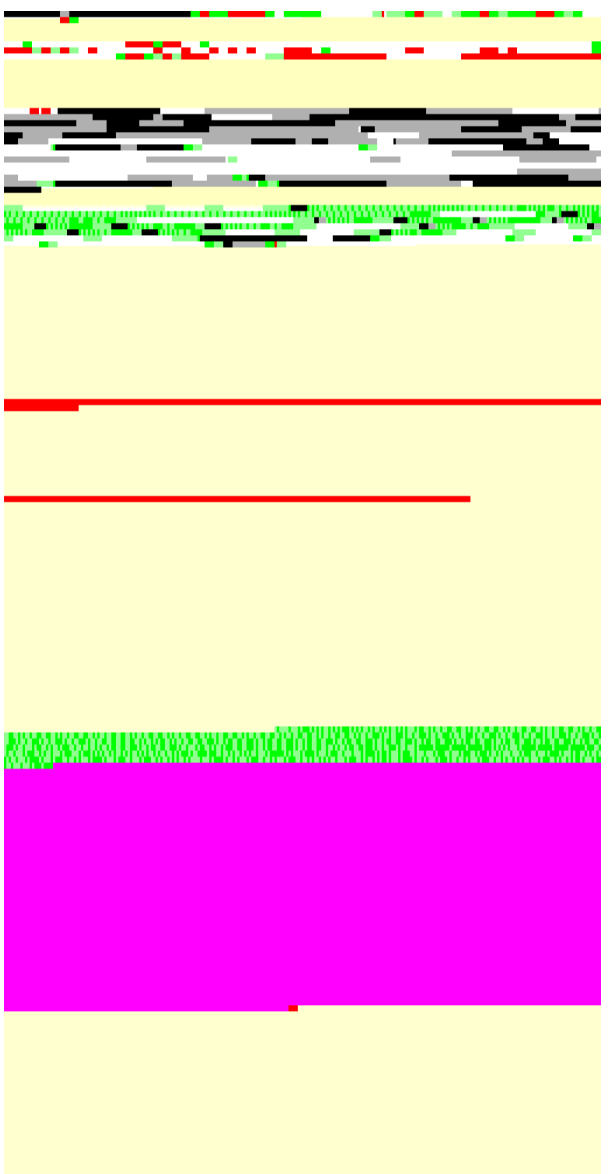
of a
32-bit-compiler knowing 7 instructions

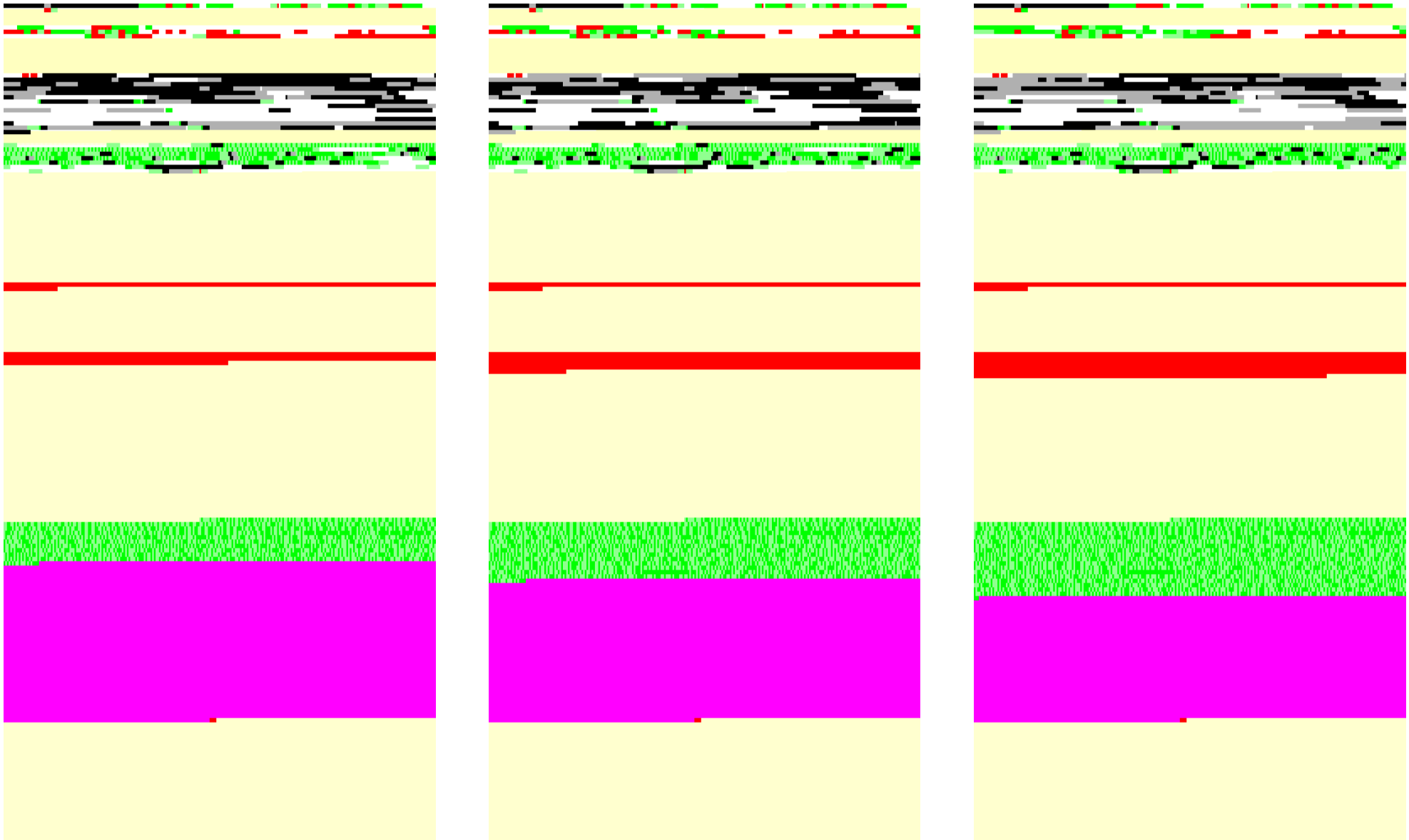
. (“ f ?? adr loko

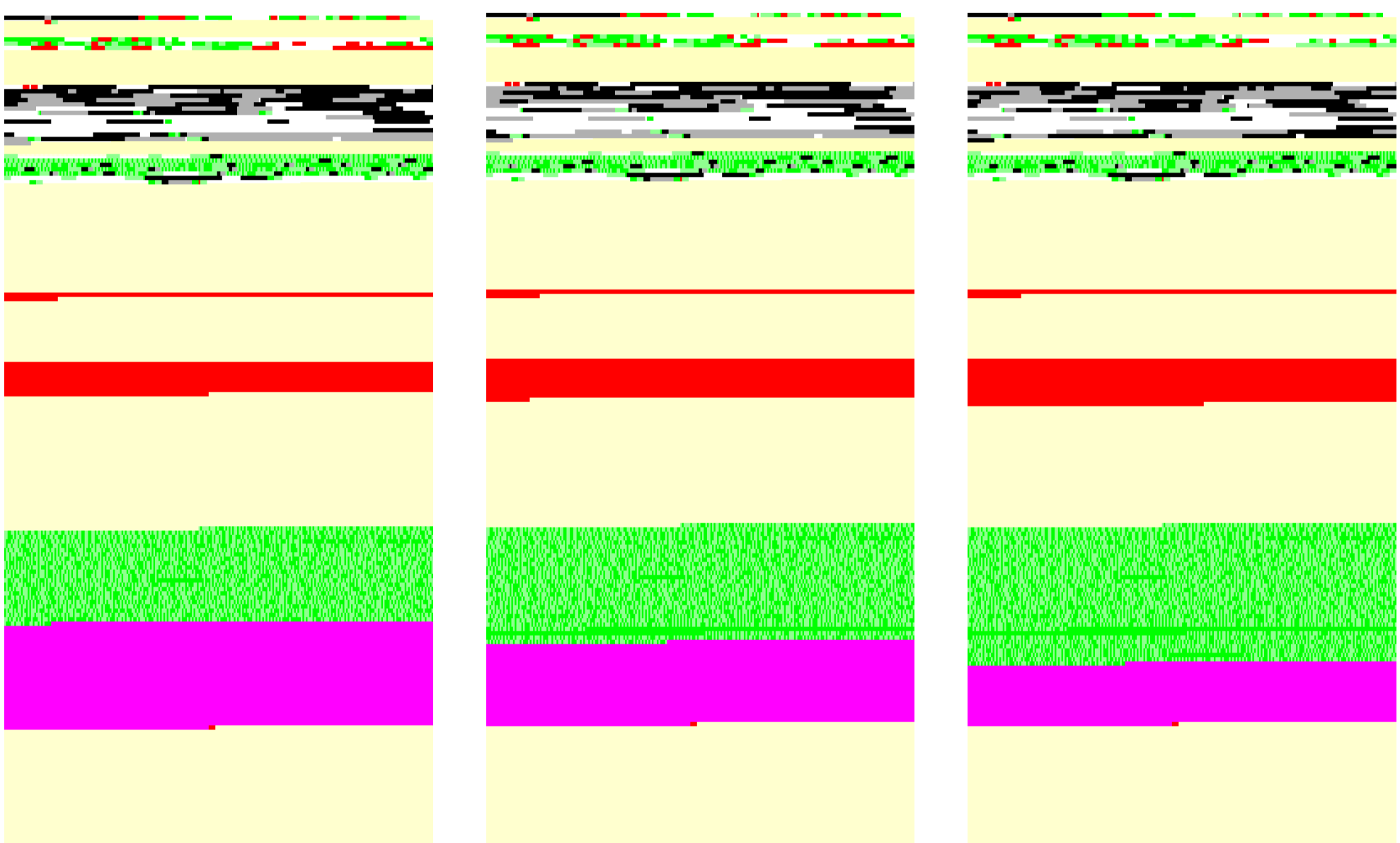
Compiled with
32-bit-compiler 39 instructions total

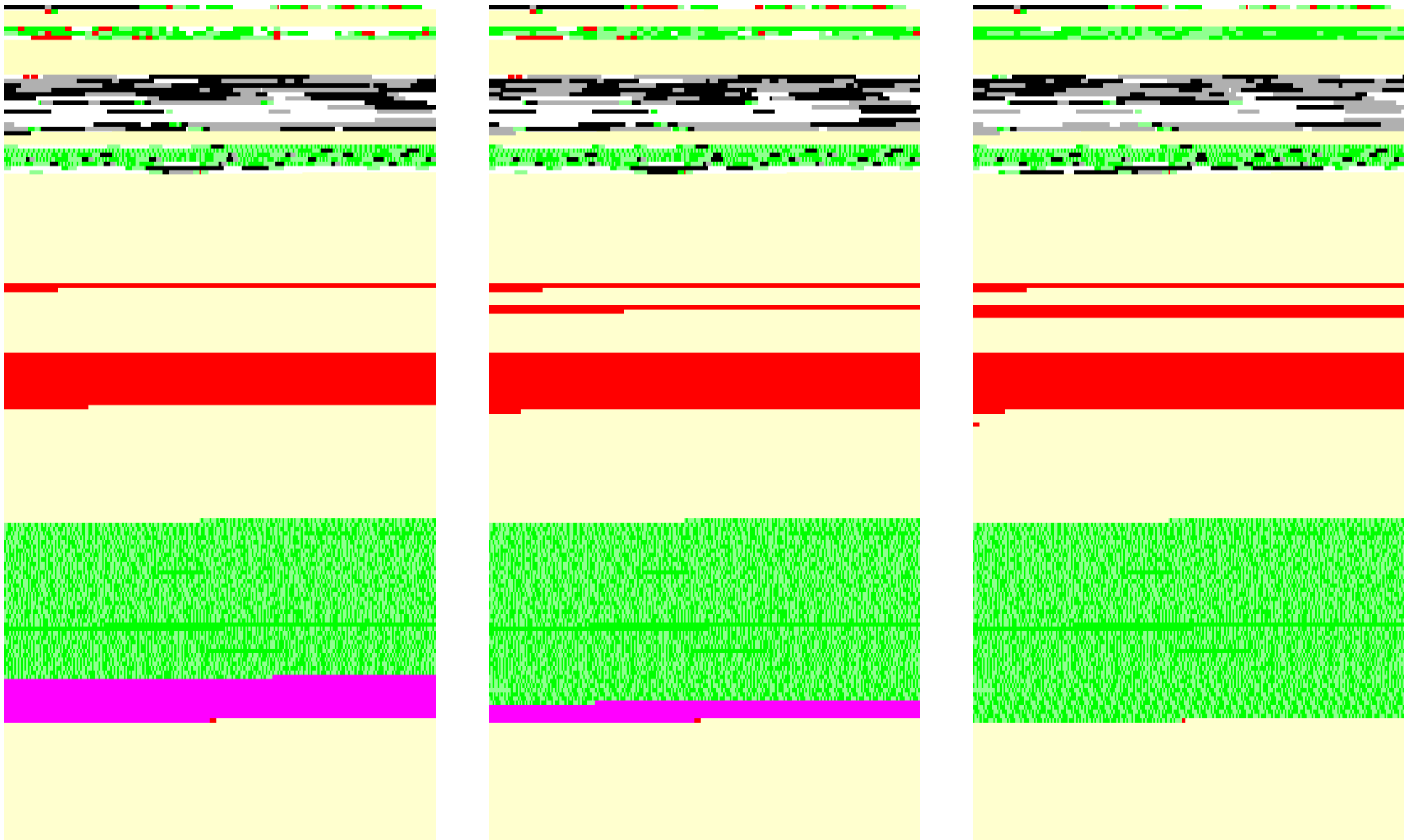
. (“ f ?? adr loko tempo t1 t2 komencu finu skribu legu a= a<- a-> a+ a- iru ir0 irt irs re irs-c a=c c=a a<>c a->c c-> a a+c a-c dump origino kreu nombro vico grandeco bf=

Info about my Esperanto programming language „E“
download of 32-bit compiler for Linux & Windows
See http://uli.esperanto-muenchen.de/e-esp.htm







Comparing the Performance of Computers

Measuring the compilation time to compile the 64-bit compiler
using various computers and dividing the results by the number of instructions,
I got instructions per second.
Thus I got an idea of the performance of these computers.

Computer	Compiler	instructions	compilation time µs	billion instr./sec
1	64-bit	79398	40	1.98
1	32-bit	828560	309	2.68
2	32-bit	828560	832	1.00
3	32-bit	828560	717	1.16
4	32-bit	828560	3756	0.22

1) Linux Knoppix 9.3 HP- Laptop Celeron N4000 @1,1GHz 2 Cores 2 threads 8GB ram 2488 Mhz cache size 4096 KB level 1 2x24KB data 2x32KB instr

2) Linux Knoppix 8.3 EEE PC 900 Celeron M processor 900 Mhz 1GB ram 900 MHz cache size 512 KB level 1 32KB data 32KB instr

3) Linux 3.14.0+ Tablet Odys Nova7 Android 6.0.1 2016 Genuine Intel Model 93 @728 4 Cores 1040 MHz cache size 512 KB

4) Linux 2.4.29 Puppy AMD K6 300 MHz 94 MB ram 300 MHz cache size 64 KB

Comparing the Performance of Compilers

Measuring the performance of compilers I compiled with various compilers and sources as noted below

Compiler	Compiler compiled	source	instructions	compilation time µs	billion instr./sec
1	3	A	7351738	150821	0.049
2	3	A	9592270	3640 … 4278	2.635 … 2.242
3	3	A	5448488	2126 … 3426	2.562 … 1.590
3	3	B	828560	287 … 335	2.886 … 2.473
4	4	C	79398	38 … 40	2.089 ... 1.985

1 32-bit compiler with 39 instructions code & data in the same memory block ==> more than 1,500,000 cache fails !!!

2 32-bit compiler with 39 instructions code& data separated only 4 cache fails instructions not orderd by number of appearance in source

3 32-bit compiler same compiler as in 2 but instructions in library ordered ==> the compiler finds the instructions quickly without need to search the whole library

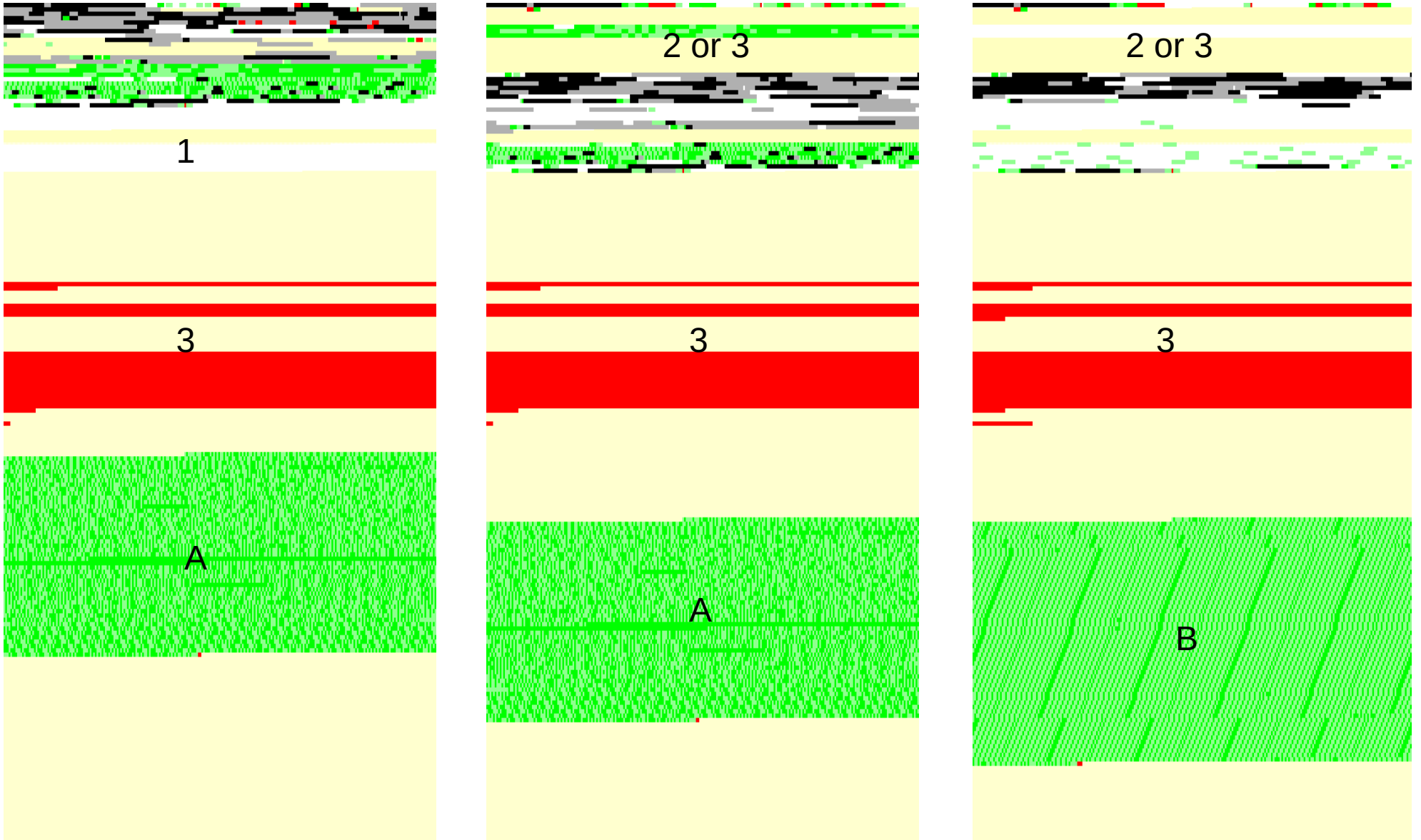
4 64-bit compiler with only 1 instruction code & data separated

A source for 32-bit compiler with 7 instructions code & data separated written in „Esperanto programming language (E)“

B source for 32-bit compiler with 7 instructions code & data separated using mainly hex-numbers

C source for 64-bit compiler with 1 instruction code & data separated using mainly hex-numbers

Computer in all tests:
Linux Knoppix 9.3 – Celeron N4000 @1,1GHz 2 Cores 2 threads 8GB ram 2488 Mhz cache size 4096 KB level 1 caches 2x24KB for data 2x32KB for instructions


1
2 or 3
2 or 3
3
3
3
A
A
B

e-64

La plej malgranda kaj tamen etendigebla 64-bita kompililo

english translations
---

initialization

process one word

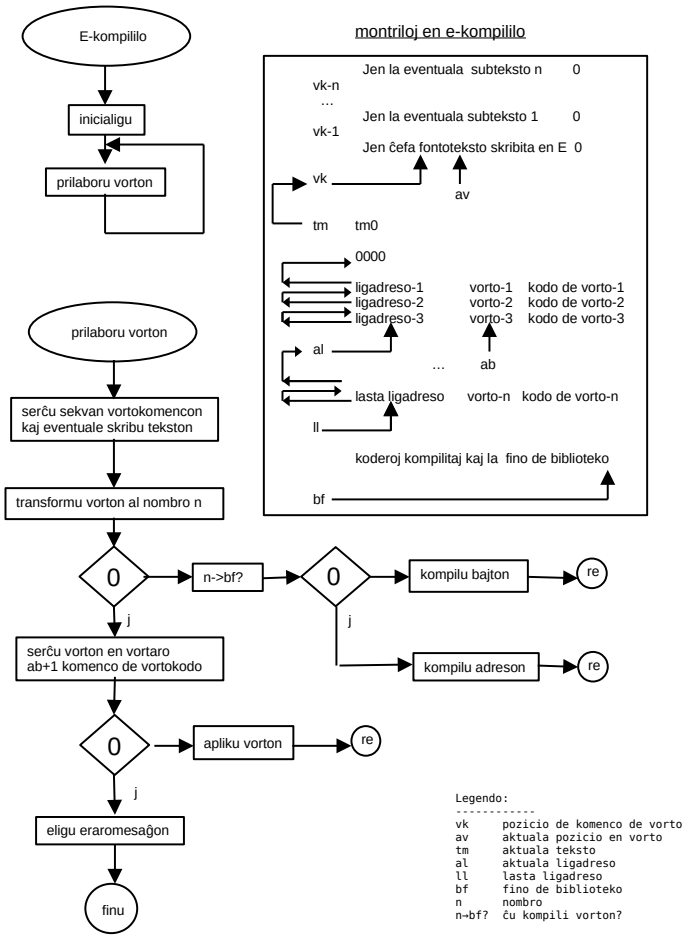
look for word start or
Write source

transform word to number

look for word in library

applicate the word

output error



english translations
---

pointers inside the compiler

here the subtexts

here is the main sourcetext

last linkaddress word-n code of word-n

codes compiled and the end of library

compile byte

compile address

Legend:
vk position of word start
av actual position inside word
tm actual text
al actual linkaddress
ll last linkaddresss
bf end of library
n number
n→bf? compile a byte

Hexdump of the extracted Baby-Compiler

```
00000000  7f 45 4c 46 02 01 01 00  00 45 73 70 65 72 6f 20  |.ELF.....Espero |
00000010  02 00 3e 00 01 00 00 00  d7 11 40 00 00 00 00 00  |..>.......@.....|
00000020  40 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |@...............|
00000030  00 00 00 00 40 00 38 00  01 00 00 00 00 00 00 00  |....@.8.........|
00000040  01 00 00 00 07 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 40 00 00 00 00 00  00 00 40 00 00 00 00 00  |..@.......@.....|
00000060  00 00 00 50 01 00 00 00  00 00 00 50 01 00 00 00  |...P.......P....|
00000070  00 10 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000120  00 00 00 00 00 00 00 00  00 01 40 00 00 00 00 00  |..........@.....|
00000130  6b 12 40 00 00 00 00 00  00 00 00 00 00 00 00 00  |k.@.............|
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000150  00 10 00 00 00 00 00 00  63 12 40 00 00 00 00 00  |........c.@.....|
00000160  bc 0a 00 00 00 00 00 00  23 01 00 00 00 00 00 00  |........#.......|
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000180  10 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000190  01 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
000001a0  65 72 61 72 6f 21 0a 00  00 00 00 00 00 00 00 00  |eraro!..........|
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  00 00 00 00 00 00 00 00  31 00 00 00 00 00 00 00  |........1.......|
00000410  03 e3 7f 66 00 00 00 00  75 cf 01 00 00 00 00 00  |...f....u.......|
00000420  03 e3 7f 66 00 00 00 00  a6 cf 01 00 00 00 00 00  |...f............|
00000430  88 ff ff ff 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000440  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

```
00001000  4d 8b 18 49 ff c3 41 8a  1b 80 fb 21 73 f5 41 8a  |M..I..A....!s.A.|
00001010  1b 80 fb 00 74 0e 80 fb  21 73 05 49 ff c3 eb ee  |....t...!s.I....|
00001020  4d 89 18 c3 48 b8 00 01  40 00 00 00 00 00 4c 29  |M...H...@.....L)|
00001030  c0 75 4d 48 8b 04 25 30  01 40 00 48 05 95 2d 00  |.uMH..%0.@.H..-.|
00001040  00 49 89 00 48 89 04 25  48 01 40 00 b8 00 00 00  |.I..H..%H.@.....|
00001050  00 bf 00 00 00 00 48 8b  14 25 50 01 40 00 48 8b  |......H..%P.@.H.|
00001060  34 25 48 01 40 00 0f 05  48 8b 1c 25 48 01 40 00  |4%H.@...H..%H.@.|
00001070  48 01 c3 b8 00 00 00 00  48 89 03 4d 8b 18 eb 8e  |H.......H..M....|
00001080  49 83 e8 08 e9 77 ff ff  ff 4d 8b 18 4c 8b 2c 25  |I....w...M..L.,%|
00001090  58 01 40 00 4d 89 ee 4d  89 df 49 ff cf 49 83 c6  |X.@.M..M..I..I..|
000010a0  08 41 8a 0e 80 f9 21 72  1a 49 ff c7 41 8a 1f 38  |.A....!r.I..A..8|
000010b0  d9 74 21 4d 8b 6d 00 49  83 fd 00 75 d7 b8 00 00  |.t!M.m.I...u....|
000010c0  00 00 c3 49 ff c7 41 8a  1f 80 fb 21 73 e5 b8 01  |...I..A....!s...|
000010d0  00 00 00 c3 49 ff c6 eb  c8 b8 00 00 00 00 48 89  |....I.........H.|
000010e0  04 25 70 01 40 00 49 8b  00 48 89 04 25 40 01 40  |.%p.@.I..H..%@.@|
000010f0  00 48 8b 04 25 40 01 40  00 8a 18 80 eb 30 72 6f  |.H..%@.@.....0ro|
00001100  80 fb 0a 72 0d 80 eb 11  72 65 80 fb 18 73 60 80  |...r....re...s`.|
00001110  c3 0a 88 1c 25 88 01 40  00 8a 04 25 80 01 40 00  |....%..@...%..@.|
00001120  28 c3 74 4b 48 8b 04 25  70 01 40 00 48 0f af 04  |(.tKH..%p.@.H...|
00001130  25 80 01 40 00 48 03 04  25 88 01 40 00 48 89 04  |%..@.H..%..@.H..|
00001140  25 70 01 40 00 48 8b 04  25 40 01 40 00 48 ff c0  |%p.@.H..%@.@.H..|
00001150  48 89 04 25 40 01 40 00  8a 18 80 fb 21 72 02 eb  |H..%@.@.....!r..|
00001160  90 4c 8b 14 25 70 01 40  00 b8 01 00 00 00 c3 b8  |.L..%p.@........|
00001170  00 00 00 00 c3 49 ff c6  41 ff d6 c3 b8 01 00 00  |.....I..A.......|
00001180  00 bf 01 00 00 00 ba ff  ff 00 00 48 be 00 00 40  |...........H...@|
00001190  00 00 00 00 00 0f 05 b8  3c 00 00 00 bf 00 00 00  |........<.......|
000011a0  00 0f 05 48 8b 04 25 90  01 40 00 48 83 f8 01 75  |...H..%..@.H...u|
000011b0  08 45 88 14 24 49 ff c4  c3 4d 89 14 24 49 83 c4  |.E..$I...M..$I..|
000011c0  08 b8 01 00 00 00 48 89  04 25 98 01 40 00 48 89  |......H..%..@.H.|
000011d0  04 25 90 01 40 00 c3 4c  8b 1c 25 30 01 40 00 4c  |.%..@..L..%0.@.L|
000011e0  8b 24 25 30 01 40 00 4c  89 24 25 00 01 40 00 49  |.$%0.@.L.$%..@.I|
000011f0  b8 00 01 40 00 00 00 00  00 41 b9 00 00 00 00 e8  |...@.....A......|
00001200  02 00 00 00 eb f9 e8 f5  fd ff ff e8 c9 fe ff ff  |................|
00001210  48 83 f8 00 75 15 e8 6e  fe ff ff 48 83 f8 00 74  |H...u..n...H...t|
00001220  05 e9 4f ff ff ff e9 51  ff ff ff e9 73 ff ff ff  |..O....Q....s...|
00001230  00 00 00 00 00 00 00 00  30 12 40 00 00 00 00 00  |........0.@.....|
00001240  2e 20 48 8b 04 25 58 01  40 00 49 89 04 24 4c 89  |. H..%X.@.I..$L.|
00001250  24 25 58 01 40 00 49 83  c4 08 4c 89 24 25 30 01  |$%X.@.I...L.$%0.|
00001260  40 00 c3 38 12 40 00 00  00 00 00 00 00 00 00 00  |@..8.@..........|
00001270  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001400
```

The file has to be extended to 00010000 with zero-bytes!

```
                ; kompiliere mit:     nasm -f bin compiler.asm -l compiler.lst -o  64-bit-compiler
                                    BITS 64
                                         org 0x400000
                                    dump:                                       ; E L F  -  H E A D E R
00000000 7F454C4602010100                           db      0x7F, "ELF", 2, 1, 1, 0  ;   e_ident
00000008 0045737065726F20                           db      0, "Espero "
00000010 0200                                       dw      2                   ;   e_type
00000012 3E00                                       dw      0x3E                ;   e_machine
00000014 01000000                                   dd      1                   ;   e_version
00000018 [D711000000000000]                         dq      komenco             ;   e_entry
00000020 4000000000000000                           dq      0x40                ;   e_phoff
00000028 0000000000000000                           dq      0                   ;   e_shoff
00000030 00000000                                   dd      0                   ;   e_flags
00000034 4000                                       dw      0x40                ;   e_ehsize
00000036 3800                                       dw      0x38                ;   e_phentsize
00000038 0100                                       dw      1                   ;   e_phnum
0000003A 0000                                       dw      0                   ;   e_shentsize
0000003C 0000                                       dw      0                   ;   e_shnum
0000003E 0000                                       dw      0                   ;   e_shstrndx
00000040 01000000                                   dd      1                   ;   p_type
00000044 07000000                                   dd      7                   ;   p_flags
00000048 0000000000000000                           dq      0                   ;   p_offset
00000050 0000400000000000                           dq      0x400000            ;   p_vaddr
00000058 0000400000000000                           dq      0x400000            ;   p_paddr
00000060 0000005001000000                           dq      0x150000000         ;   p_filesz
00000068 0000005001000000                           dq      0x150000000         ;   p_memsz
00000070 0010000000000000                           dq      0x1000              ;   p_align

00000078 00<rept>                    align 0x100, db 0

                                                                                ; K E R N E L  V A R I A B L E S
00000100 0000000000000000           tm0:    dq      0                           ; montrilo al chefa teksto
00000108 0000000000000000                   dq      0
00000110 0000000000000000                   dq      0
00000118 0000000000000000                   dq      0
00000120 0000000000000000                   dq      0
00000128 [0001000000000000]         tm:     dq      tm0                         ; montrilo al akt. vortokomenco
00000130 [6B12000000000000]         bf:     dq      bibliotekofino
00000138 0000000000000000           vk:     dq      0                           ; steht in r8
00000140 0000000000000000           av:     dq      0                           ; steht in r11 bzw. r15
00000148 0000000000000000           enad:   dq      0
00000150 0010000000000000           enlo:   dq      0x1000
00000158 [6312000000000000]         ll:     dq      lastaligadreso
00000160 BC0A000000000000           alig:   dq      0xABC
00000168 2301000000000000           ab:     dq      0x123
```

```
45 00000170 0000000000000000             n:        dq       0
46 00000178 0000000000000000             nn:       dq       0
47 00000180 1000000000000000             b:        dq       0x10
48 00000188 0000000000000000             cif:      dq       0
49 00000190 0100000000000000             kob?:     dq       1
50 00000198 0100000000000000             flag:     dq       1
51 000001A0 657261726F210A              error:     db       "eraro!",0x0A          ; Errortext
53
55 00000800 00<rept>                       align 0x1000,db 0
56                                                                                 ; S T A R T  O F  K E R N E L
57                                         sevoko:                                 ; SEARCH START OF NEXT WORD
58 00001000 4D8B18                                 mov r11, [r8]
59 00001003 49FFC3                         l1:     inc r11                         ; montrilo al sekva signo
60 00001006 418A1B                                 mov bl, [r11]                   ; bl = (vk) == aktuelles Zeichen 2
61 00001009 80FB21                                 cmp bl, 0x21                    ; springe wenn ein Buchstabe
62 0000100C 73F5                                   jnc  l1
63 0000100E 418A1B                         l4:     mov bl, [r11]
64 00001011 80FB00                                 cmp bl, 0
65 00001014 740E                                   jz l2                           ; springe wenn Textende erreicht
66 00001016 80FB21                                 cmp bl, 0x21
67 00001019 7305                                   jnc l5                          ; springe wenn ein Buchstabe
68 0000101B 49FFC3                                 inc r11
69 0000101E EBEE                                   jmp l4                          ; springe wenn kein Buchstabe
70 00001020 4D8918                         l5:     mov [r8], r11                   ; Wortanfang gefunden Rückkehr
71 00001023 C3                                     ret
72 00001024 48B8-                          l2:     mov rax, tm0
72 00001026 [0001000000000000]
73 0000102E 4C29C0                                 sub rax, r8
74 00001031 754D                                   jnz l3
75 00001033 488B0425[30010000]                     mov rax, [bf]
76 0000103B 4805952D0000                           add rax, 0x2D95                 ; Abstand Text zu bf auf 3000 setzen
77 00001041 498900                                 mov [r8],rax
78 00001044 48890425[48010000]                     mov [enad], rax
79 0000104C B800000000                             mov rax, 0                      ; Text schreiben
80 00001051 BF00000000                             mov rdi, 0
81 00001056 488B1425[50010000]                     mov rdx, [enlo]
82 0000105E 488B3425[48010000]                     mov rsi, [enad]
83 00001066 0F05                                   syscall
84 00001068 488B1C25[48010000]                     mov rbx, [enad]                 ; am Textende 0 anhängen
85 00001070 4801C3                                 add rbx, rax                    ; rax ist Textlänge
86 00001073 B800000000                             mov rax, 0
87 00001078 488903                                 mov [rbx], rax
88 0000107B 4D8B18                                 mov r11, [r8]
89 0000107E EB8E                                   jmp l4
90 00001080 4983E808                       l3:     sub r8,8
91 00001084 E977FFFFFF                             jmp sevoko
```

```
 92
 93 00001089 4D8B18                      svev:     mov r11, [r8]                ; SEARCH WORD IN LIBRARY
 94 0000108C 4C8B2C25[58010000]                   mov r13, [ll]                ; aktueller Wortanfang        av<-r11  steht in r8
 95 00001094 4D89EE                     ls0:      mov r14, r13                 ; aktuelle Linkadresse        al<-r13
 96 00001097 4D89DF                               mov r15, r11                 ; aktuelle Pos. in Bibliothek ab=r14
 97 0000109A 49FFCF                               dec r15                      ; aktuelle Pos. im Text gleich Wortanf.
 98 0000109D 4983C608                             add r14, 0x8                 ; av=av-1
 99 000010A1 418A0E                     ls3:      mov cl,  [r14]               ; ab=ab+8 zeigt auf ersten Buchstaben
100 000010A4 80F921                               cmp cl,  0x21                ; Zeichen an Position ab
101 000010A7 721A                                 jc  ls1                      ; (ab) Buchstabe?
102 000010A9 49FFC7                               inc r15                      ;
103 000010AC 418A1F                               mov bl,[r15]                 ; av=av+1
104 000010AF 38D9                                 cmp cl, bl                   ; bl=(av)
105 000010B1 7421                                 jz  ls2                      ; ist (ab)=(av) ?
106 000010B3 4D8B6D00                  ls4:       mov r13,[r13]                ; wenn ja springen
107 000010B7 4983FD00                             cmp r13, 0                   ; wenn nein nä. Wort suchen al=(al)
108 000010BB 75D7                                 jnz  ls0                     ; ist alig = 0  also Bibliotheksanfang?
109 000010BD B800000000                           mov rax, 0                   ; wenn nein dann weiter
110 000010C2 C3                                   ret                          ; wenn ja dann Wort nicht gefunden
111 000010C3 49FFC7                    ls1:       inc r15                      ;
112 000010C6 418A1F                               mov bl,[r15]                 ; (ab) kein Buchstabe av+1
113 000010C9 80FB21                               cmp bl, 0x21                 ;
114 000010CC 73E5                                 jnc ls4                      ; (av) Buchstabe?
115 000010CE B801000000                           mov rax, 1                   ;
116 000010D3 C3                                   ret                          ; Wort in Bibliothek gefunden
117 000010D4 49FFC6                    ls2:       inc r14                      ;
118 000010D7 EBC8                                 jmp ls3                      ; ab=ab+1
```

```
119                                                                     ; TRANSFORM A WORD INTO A NUMBER
120 000010D9 B800000000                 tvan:   mov rax, 0              ; Initialisierung
121 000010DE 48890425[70010000]                 mov [n], rax
122 000010E6 498B00                             mov rax, [r8]
123 000010E9 48890425[40010000]                 mov [av], rax
124 000010F1 488B0425[40010000]         lt1:    mov rax, [av]           ; aktuelle Position im Text
125 000010F9 8A18                               mov bl, [rax]           ; aktuelles Zeichen
126 000010FB 80EB30                             sub bl, 0x30            ; wenn < 0 keine Ziffer
127 000010FE 726F                               jc  lt2
128 00001100 80FB0A                             cmp bl, 0x0A            ; wenn < 10 dann Ziffer zw. 0 u. 9
129 00001103 720D                               jc  lt3
130 00001105 80EB11                             sub bl, 0x11            ; wenn < 17 dann keine Ziffer
131 00001108 7265                               jc  lt2
132 0000110A 80FB18                             cmp bl, 0x18            ; wenn > 24 dann auch keine Ziffer
133 0000110D 7360                               jnc lt2
134 0000110F 80C30A                             add bl, 0x0A            ; 10 addieren weil cif A,B,C,D,E,F
135 00001112 881C25[88010000]           lt3:    mov [cif], bl
136 00001119 8A0425[80010000]                   mov al, [b]
137 00001120 28C3                               sub bl, al
138 00001122 744B                               jz  lt2
139 00001124 488B0425[70010000]                 mov  rax, [n]           ;  n=n*b+cif
140 0000112C 480FAF0425-                        imul rax, [b]
140 00001131 [80010000]
141 00001135 48030425[88010000]                 add  rax, [cif]
142 0000113D 48890425[70010000]                 mov  [n], rax
143 00001145 488B0425[40010000]                 mov  rax, [av]          ; av=av+1
144 0000114D 48FFC0                             inc  rax
145 00001150 48890425[40010000]                 mov  [av],rax
146 00001158 8A18                               mov  bl, [rax]          ; nächstes Zeichen holen
147 0000115A 80FB21                             cmp  bl, 0x21
148 0000115D 7202                               jc   lt4                ; wenn kein Buchstabe oder Ziffer
149 0000115F EB90                               jmp  lt1                ; sonst Zeichen in Ziffer wandeln
150 00001161 4C8B1425[70010000]         lt4:    mov r10, [n]
151 00001169 B801000000                         mov rax, 1              ; ist eine Zahl!
152 0000116E C3                                 ret
153 0000116F B800000000                 lt2:    mov rax, 0              ; ist keine Zahl!
154 00001174 C3                                 ret
```

```
155                                                     ; APPLICATE A WORD
156 00001175 49FFC6                    apvo:   inc r14                 ; ab=ab+1 irs ab
157 00001178 41FFD6                            call r14
158 0000117B C3                                ret
159                                                     ; OUTPUT ERROR MESSAGE
160 0000117C B801000000                 eler:   mov rax, 1              ; Memory-Dump als Error-Meldung
161 00001181 BF01000000                         mov rdi, 1
162 00001186 BAFFFF0000                         mov rdx, 0xFFFF
163 0000118B 48BE-                              mov rsi, dump
163 0000118D [0000000000000000]
164 00001195 0F05                               syscall
165 00001197 B83C000000                         mov rax, 0x3C           ; Programmm beenden
166 0000119C BF00000000                         mov rdi, 0
167 000011A1 0F05                               syscall
168
169 000011A3 488B0425[90010000]         kob:    mov rax, [kob?]         ; COMPILE BYTE
170 000011AB 4883F801                           cmp rax, 1
171 000011AF 7508                               jnz koa
172 000011B1 45881424                           mov [r12], r10b         ; 1 Byte schreiben
173 000011B5 49FFC4                             inc r12                 ; Ende-Zeiger inkrementieren
174 000011B8 C3                                 ret
175 000011B9 4D891424                   koa:    mov [r12], r10          ; COMPILE ADDRESS
176 000011BD 4983C408                           add r12, 8
177 000011C1 B801000000                         mov rax, 1
178 000011C6 48890425[98010000]                 mov [flag], rax         ; Flag und kob? auf 1 setzen
179 000011CE 48890425[90010000]                 mov [kob?], rax
180 000011D6 C3                                 ret
181
182 000011D7 4C8B1C25[30010000]         komenco:mov r11, [bf]           ; INITIALIZATION OF KERNEL
183 000011DF 4C8B2425[30010000]                 mov r12, [bf]
184 000011E7 4C892425[00010000]                 mov [tm0], r12          ; Textanfang auf bibliotekofino setzen
185 000011EF 49B8-                              mov r8, tm0
185 000011F1 [0001000000000000]
186 000011F9 41B900000000                       mov r9, 0
187
188 000011FF E802000000                 schleife: call pv               ; MAIN LOOP
189 00001204 EBF9                               jmp  schleife
190 00001206 E8F5FDFFFF                 pv:     call sevoko             ; nächsten Wortanfang suchen
191 0000120B E8C9FEFFFF                 psv:    call tvan               ; Wort in Zahl umwandeln
192 00001210 4883F800                           cmp  rax, 0
193 00001214 7515                               jnz  lk1
194 00001216 E86EFEFFFF                         call svev               ; Wort in Bibliothek suchen
195 0000121B 4883F800                           cmp  rax, 0
196 0000121F 7405                               jz   lk2
197 00001221 E94FFFFFFF                         jmp apvo                ; Wort ausführen
198 00001226 E951FFFFFF                 lk2:    jmp eler                ; Errormeldung ausgeben
199 0000122B E973FFFFFF                 lk1:    jmp kob                 ; ein Byte an das Ende kompilieren
```

```
200
201 00001230 0000000000000000        lig0:          dq 0                          ; E N D  O F  K E R N E L  -  S T A R T  O F  L I B R A R Y
202 00001238 [3012000000000000]      lig1:          dq lig0
203 00001240 2E20                                   db ". "                       ; instruction "." make a new entry in library
204 00001242 488B0425[58010000]                     mov rax, [ll]                 ; ll an bf dann [ll]=bf  bf+8
205 0000124A 49890424                               mov [r12],rax
206 0000124E 4C892425[58010000]                     mov [ll], r12
207 00001256 4983C408                               add r12, 8
208 0000125A 4C892425[30010000]                     mov [bf], r12                 ; r12 nach bf abspeichern
209 00001262 C3                                     ret
210
211 00001263 [3812000000000000]      lastaligadreso: dq lig1                      ; last link address
212                                  bibliotekofino:                              ; E N D  O F  L I B R A R Y
213 0000126B 00                                     db 0x00
```