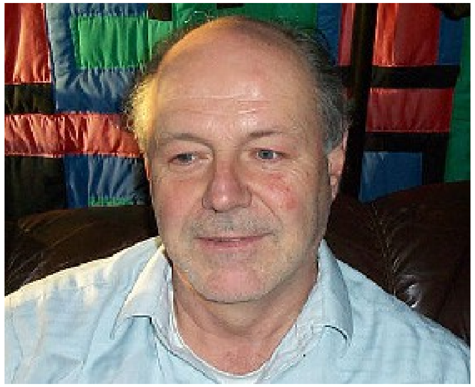
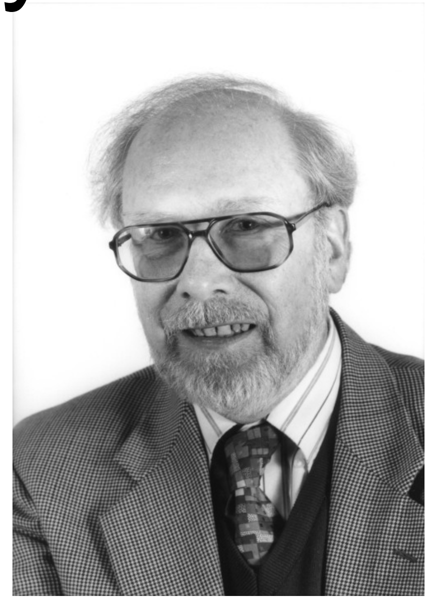
Forth Day 2013

FLOS

Stefan Mauerhofer
Switzerland

Influenced by





Chuck Moore

„Keep it simple, stupid (KISS)“

Niklaus Wirth

„Software gets slower faster than hardware gets faster“

My first Computer

1 Mhz 6502 8 bit Processor
32 KiB RAM
160 kB Floppy

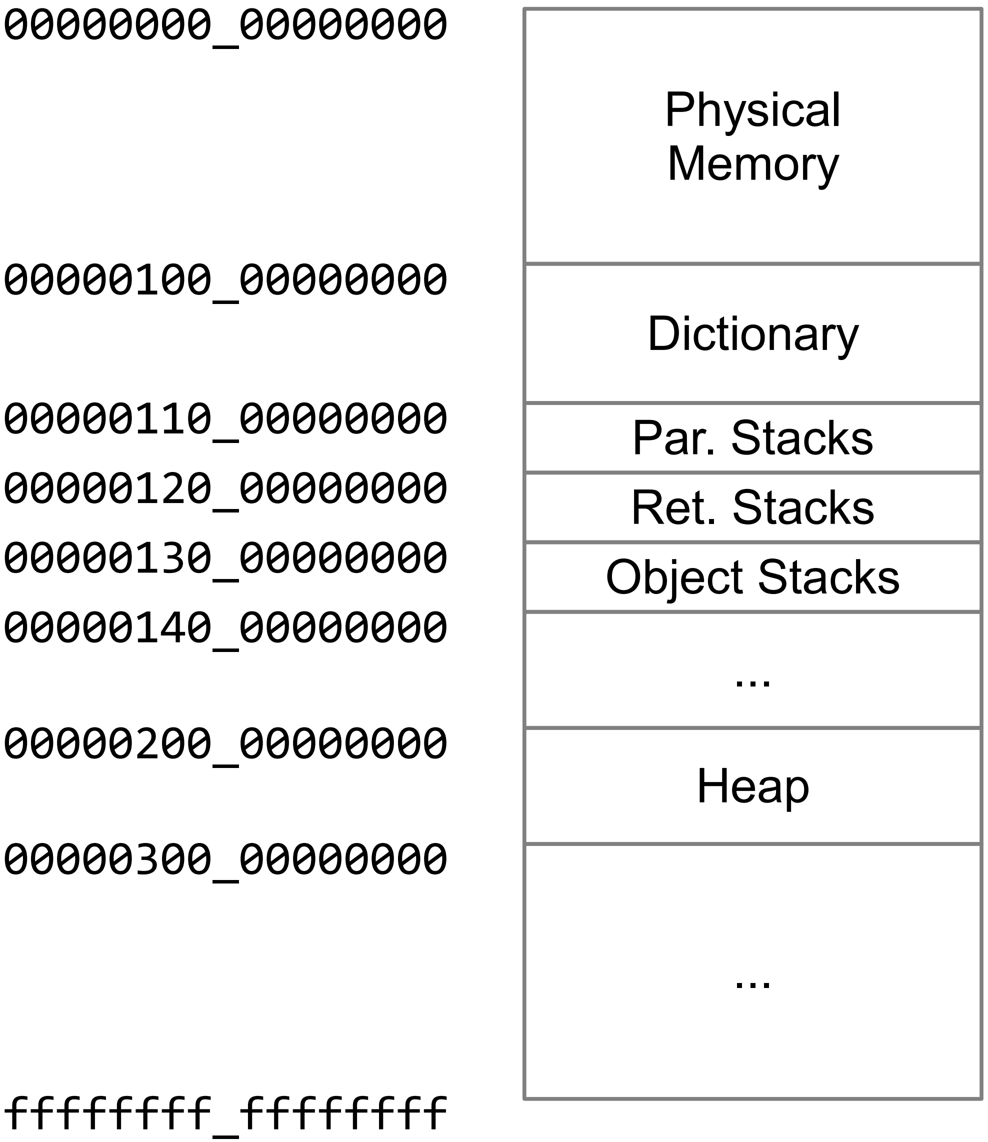
Motivation for FLOS

- Curiosity about PC architecture
- Interest in Intel 64 bit architecture
- OS independance
- Direct access to hardware
- Midlife crisis

FLOS

- Native system
- Indirect threaded code
- Object oriented pogramming
- Optimizing compiler
- 64 bit Intel processor only
- Low memory profile (< 4 MiB)
- Open system (all memory and I/O ports accessible)
- Real mode x86 emulator for BIOS
- Dynamic memory management

Memory Map



00000000_00000000

Physical Memory

00000100_00000000

Dictionary

00000110_00000000

Par. Stacks

00000120_00000000

Ret. Stacks

00000130_00000000

Object Stacks

00000140_00000000

...

00000200_00000000

Heap

00000300_00000000

...

ffffffff_ffffffff

What is Further

- Forth as a philosophy, not a standard
- Further = Forth + a little more + a little different
- Namespaces instead of vocabularies
- Built-in support for structures
- Built-in support for object orientation
- >= 2 stacks
- 32-bit ITC
- Optimizing compiler
- Block size is 4 KiB

Namespace

- Namespaces can be nested (hierarchical) and can be derived from other NS

Syntax:

```
<parent-namespace> namespace: <name>
...
;namespace
```

Namespace (example)

```
0 namespace: Foo
    0 namespace: Bar
        : myword ... ;
        myword \ directly accessible
    ;namespace
    Bar|myword \ 1 namespace prefix
;namespace
Foo|Bar|myword \ 2 namespace prefix
```

Structures

- Structure = Namespace + Data member
- No of data member must be defined a priori

Syntax:

```
<# of data member (cell)>

<parent-namespace> struct: <name>

...

;struct
```

Structures (Example)

```
2 0 struct: Point
    member ^x
    member ^y
    : distance ( point - n )
        dup ^x @ dup * ( point x^2 )
        swap ^y @ dup * ( x^2 y^2 )
        + Math|sqrt
    ;
;struct
```

Object

- Class = Structure + Method
- No of methods must be defined a priori

Syntax:

```
<# of methods>

<# of data member (cell)>

<parent-class> class: <name>

...

;class
```

Object (Example)

```
1 2 Object class: Point
    member ^x
    member ^y
    method: distance ( - n )
        self ^x @ dup * ( x^2 )
        self ^y @ dup * ( x^2 y^2 )
        + Math|sqrt
    ;method
;class
```

Dictionary

- 4 GiB size

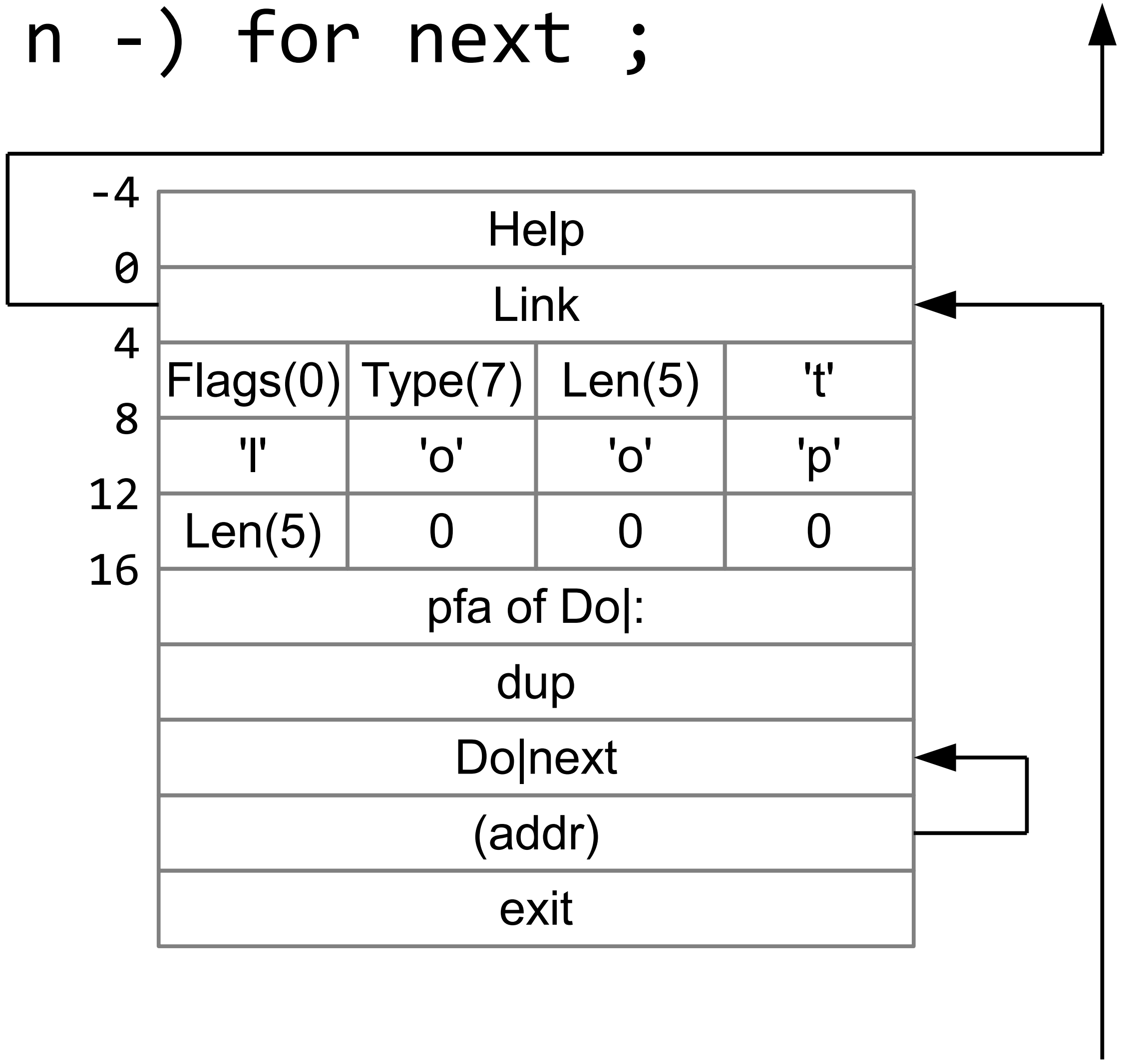
- 32 bit addressing

- 32 bit ITC

- Code page sharing with boot kernel

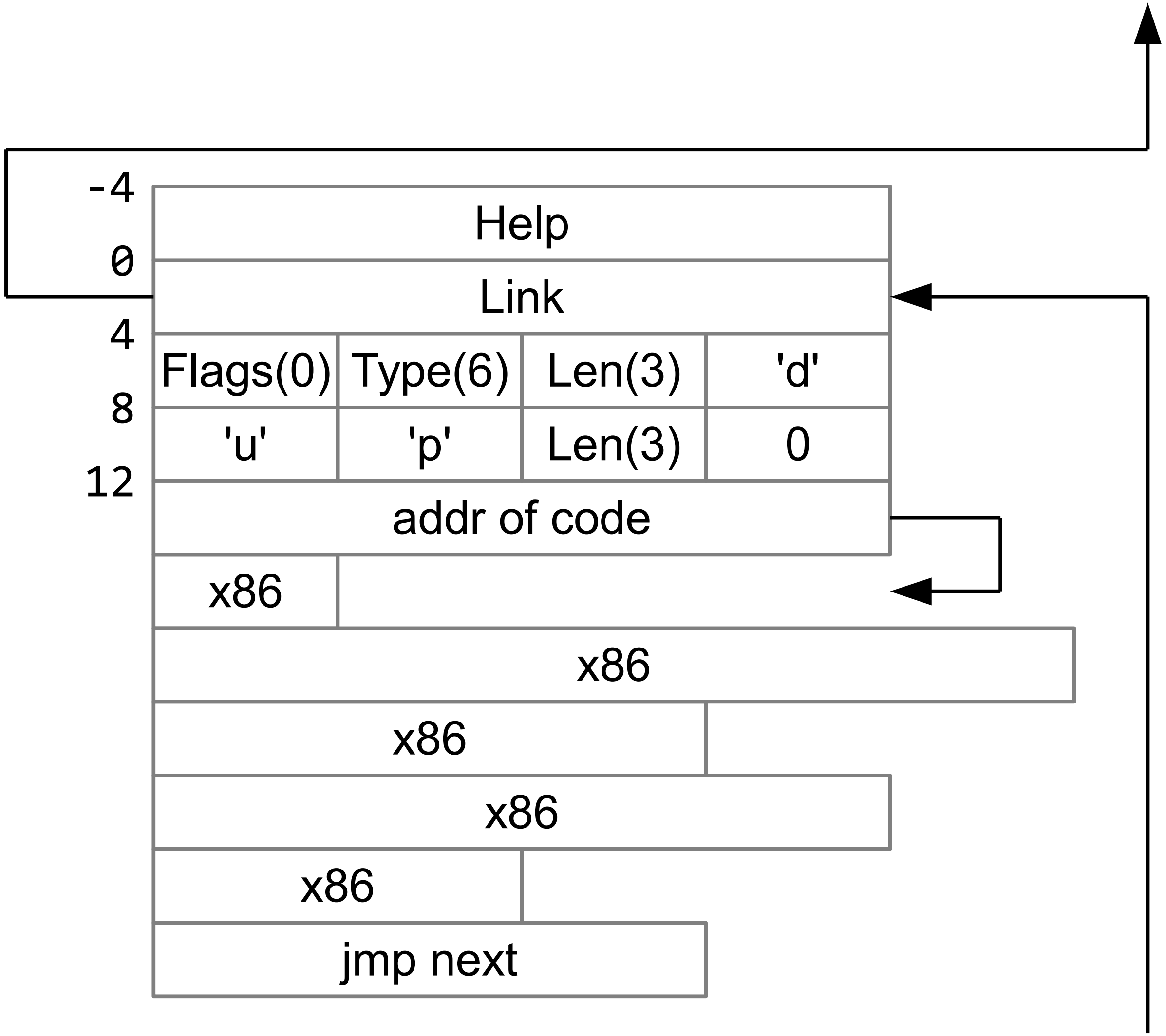
Word

```
: tloop ( n -) for next ;
```



<table>
<tr><td>-4</td><td colspan="4">Help</td></tr>
<tr><td>0</td><td colspan="4">Link</td></tr>
<tr><td>4</td><td>Flags(0)</td><td>Type(7)</td><td>Len(5)</td><td>'t'</td></tr>
<tr><td>8</td><td>'l'</td><td>'o'</td><td>'o'</td><td>'p'</td></tr>
<tr><td>12</td><td>Len(5)</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>16</td><td colspan="4">pfa of Do|:</td></tr>
<tr><td></td><td colspan="4">dup</td></tr>
<tr><td></td><td colspan="4">Do|next</td></tr>
<tr><td></td><td colspan="4">(addr)</td></tr>
<tr><td></td><td colspan="4">exit</td></tr>
</table>

Primitive Word



-4 Help

0 Link

4 Flags(0) Type(6) Len(3) 'd'

8 'u' 'p' Len(3) 0

12 addr of code

x86

x86

x86

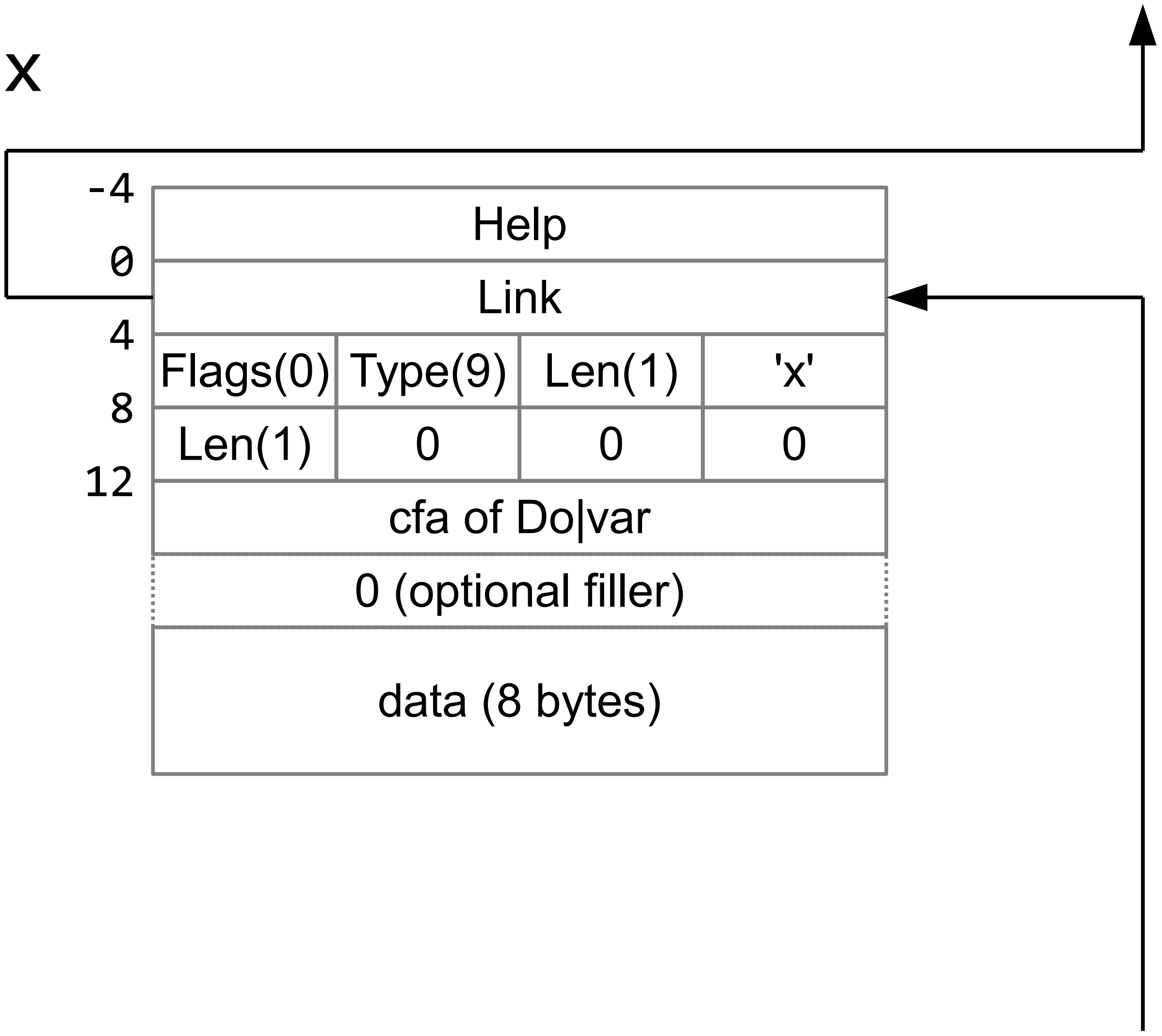
x86

x86

jmp next

Variable

```
variable x
```



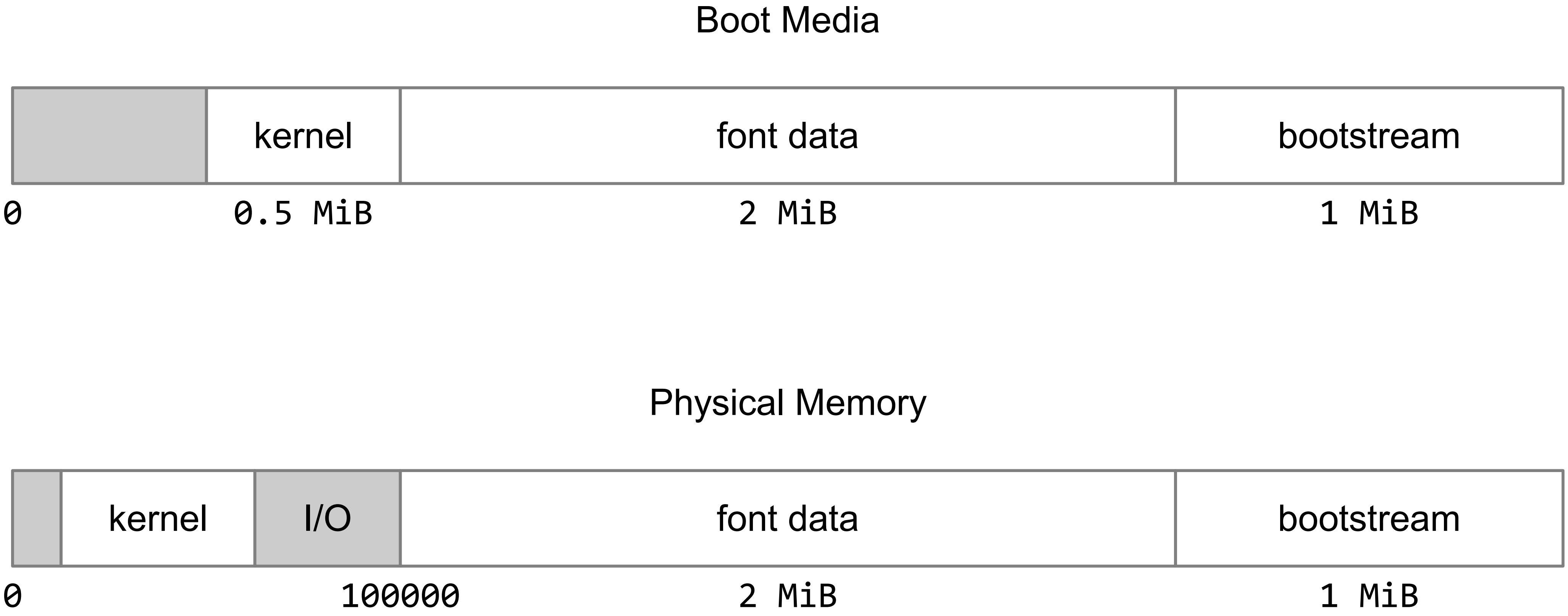
-4	Help			
0	Link			
4	Flags(0)	Type(9)	Len(1)	'x'
8	Len(1)	0	0	0
12	cfa of Do|var			
	0 (optional filler)			
	data (8 bytes)			

Booting

- First stage does mbr and/or vbr booting
- Second stage loads 3.5 MiB (0.5 MiB kernel code, 1 MiB bootstream, 2 MiB font data)
- Third stage switches to long mode (64 bit) and starts kernel
- Kernel initializes processor, memory management, basic devices (keyboard, timer, video) and start interpreting boot stream
- BIOS is still reachable in long mode through 32 bit emulator

Boot Memory Configuration

- BIOS is used to load data into memory (before switching to long mode)



Inner Interpreter

- All addresses are 32 bit offsets into dictionary space
- Indirect threaded code (ITC)
- Address of NEXT in a register
- TOS in a register
- SELF (for OOP) in a register

Register Usage

- 8 64bit legacy register
- 8 new 64bit register
- Only 2 segment register

CPU Register	
W	rax
TOS	rbx
unused, scratch	rcx
unused, scratch	rdx
IP	rsi
addr of next	rdi
RSP	rbp
PSP	rsp
unused, preserved	r8
unused, preserved	r9
unused, preserved	r10
unused, preserved	r11
unused, preserved	r12
unused, preserved	r13
DB (dictionary base)	r14
SELF	r15

UP	fs
unused	gs

Optimizing Compilier

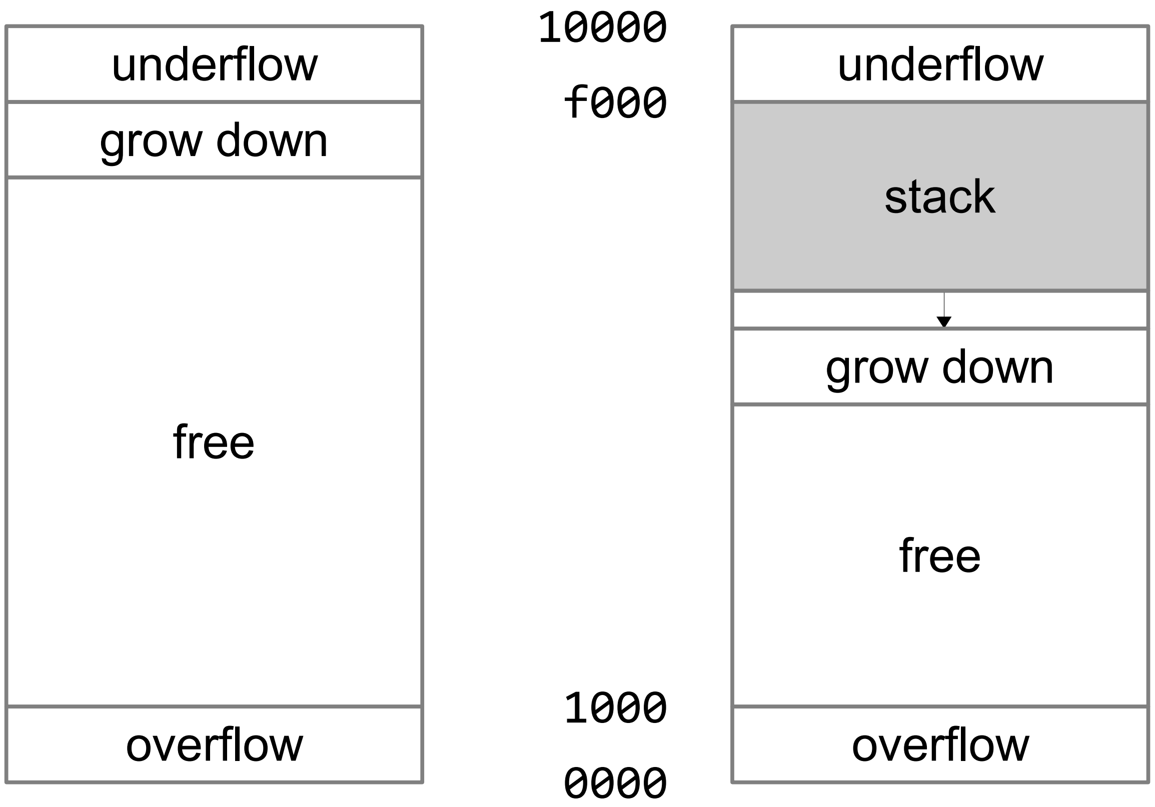
- Pattern matching and replacement
- Decreased code size
- Increased performance
- Tail recursion optimization

Stack

- Parameter & return stack
- Object stack (holds object, not references)
- More stacks possible
- Limit to 56 KiB per stack
- MMU support (overflow, underflow)

Stack (64 KiB)

- MMU support



Object Stack

- Object stack is a parameter stack for objects
- Droping an object automatically deletes it
- Independent of parameter stack

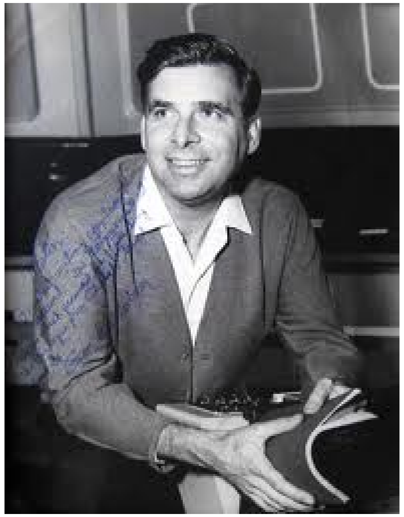
Exception Handling

- Exception frames on object stack
- Automatic object destruction on object stack
- Exception stack deletes automatically objects on the object stack

GUI

- Anti aliasing graphic primitives
- Vectorgraphic
- Simple design
- Existing design

Chuck meets Gene



Chuck Moore

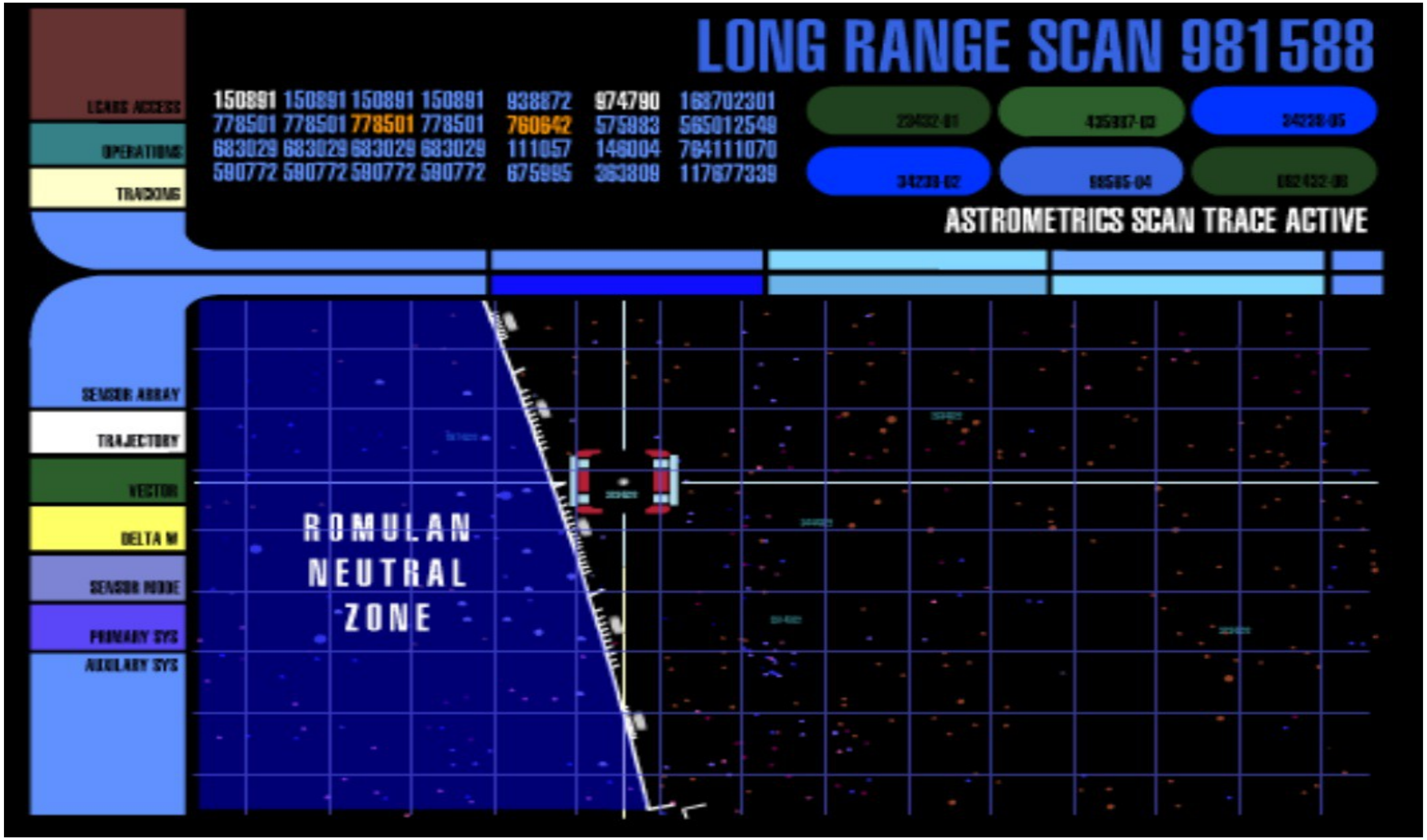
Gene Roddenberry

&

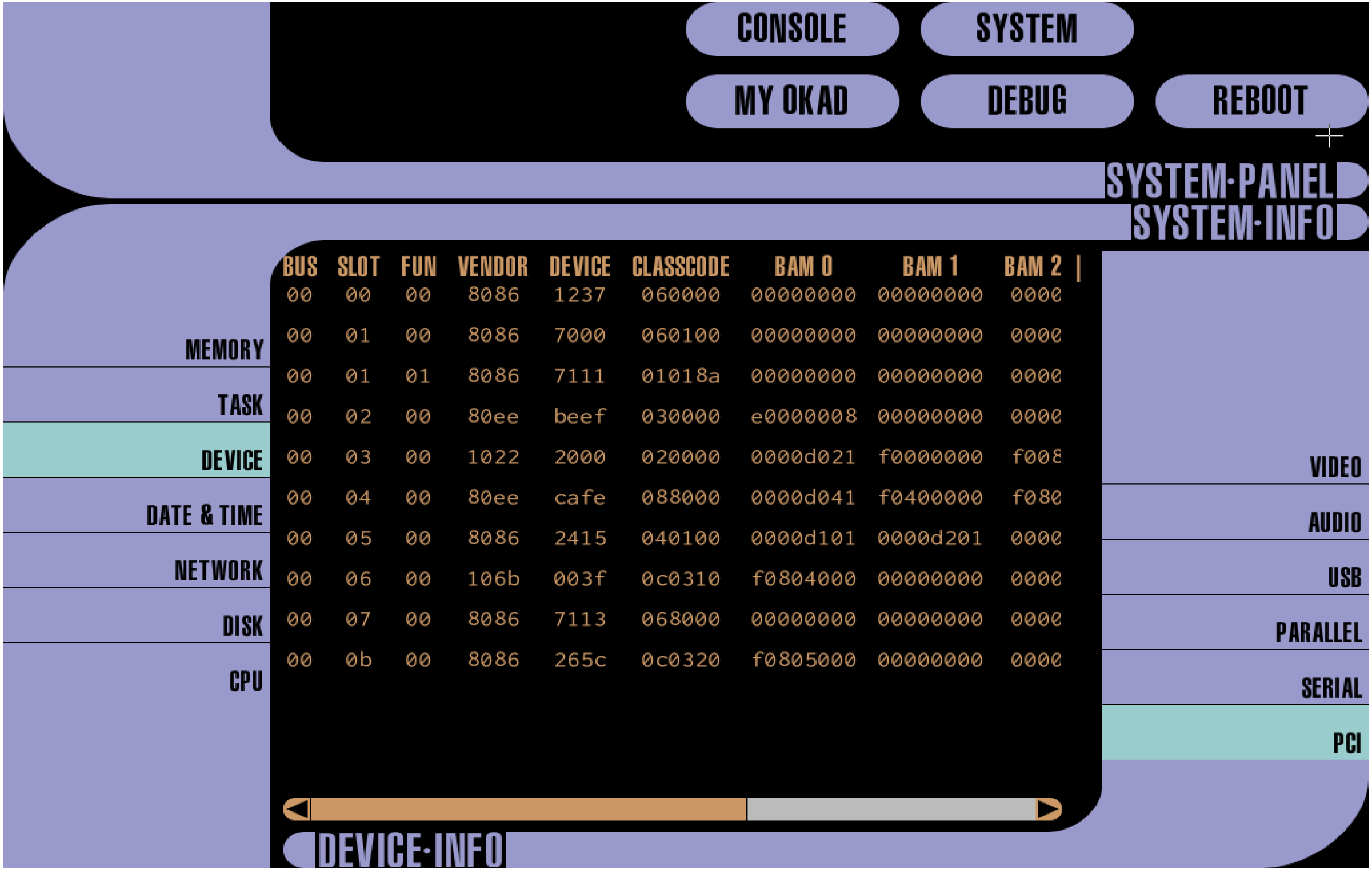
Michael Okuda

LCARS

Library Computer Access/Retrieval System



FLOS LCARS



BUS	SLOT	FUN	VENDOR	DEVICE	CLASSCODE	BAM 0	BAM 1	BAM 2
00	00	00	8086	1237	060000	00000000	00000000	0000
00	01	00	8086	7000	060100	00000000	00000000	0000
00	01	01	8086	7111	01018a	00000000	00000000	0000
00	02	00	80ee	beef	030000	e0000008	00000000	0000
00	03	00	1022	2000	020000	0000d021	f0000000	f008
00	04	00	80ee	cafe	088000	0000d041	f0400000	f080
00	05	00	8086	2415	040100	0000d101	0000d201	0000
00	06	00	106b	003f	0c0310	f0804000	00000000	0000
00	07	00	8086	7113	068000	00000000	00000000	0000
00	0b	00	8086	265c	0c0320	f0805000	00000000	0000

Future

- PC integration (x86 emulator, multi core, ACPI, ...)
- LISP integration
- GUI is not completed yet
- USB support
- Documentation