scal.cs.uni-salzburg.at
concurrent data structures

scalloc.cs.uni-salzburg.at
concurrent memory allocator

selfie.cs.uni-salzburg.at

Teaching Computer Science Through Self-Referentiality

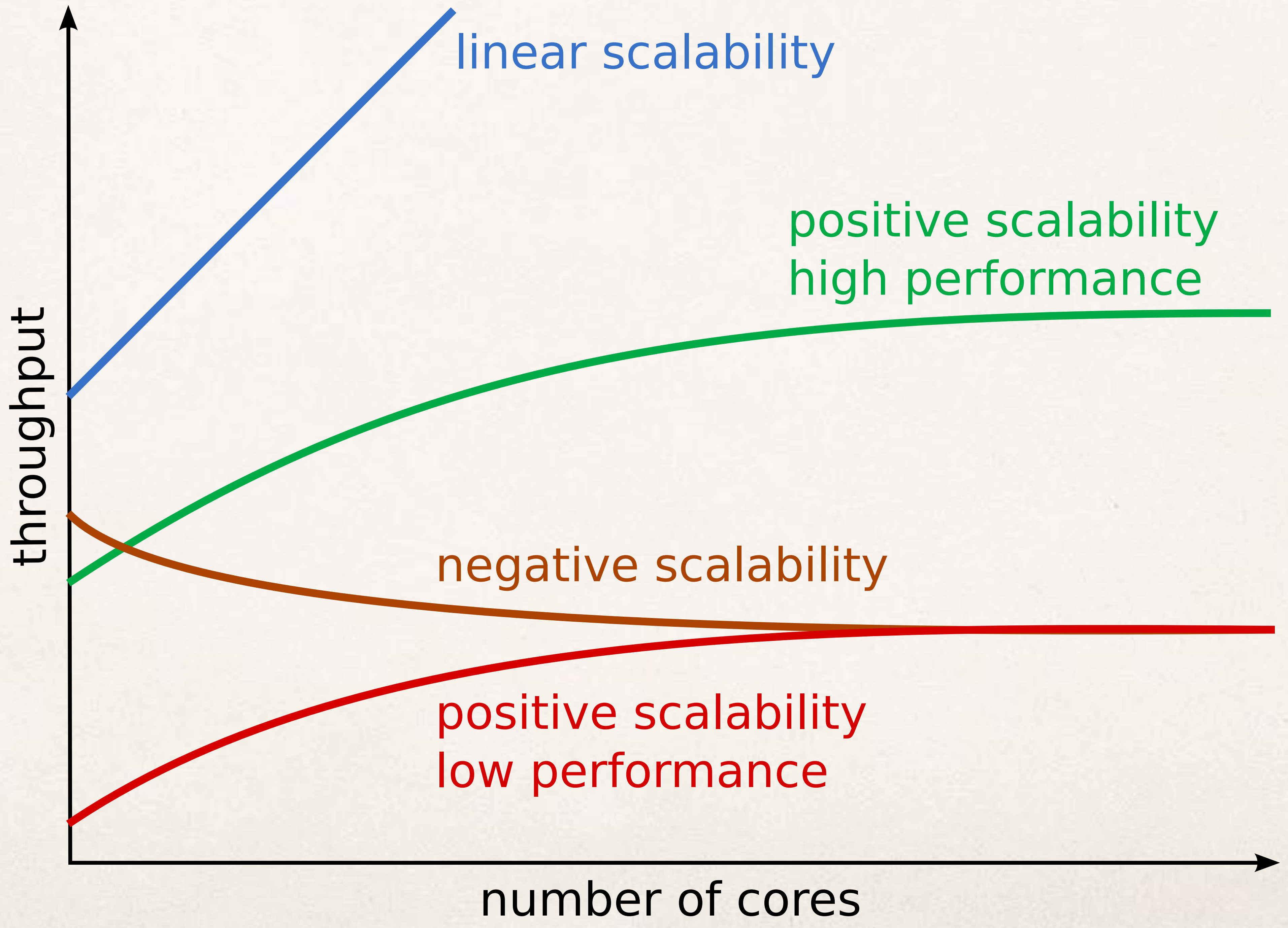
Christoph Kirsch, University of Salzburg, Austria

Hong Kong University of Science and Technology, January 2017

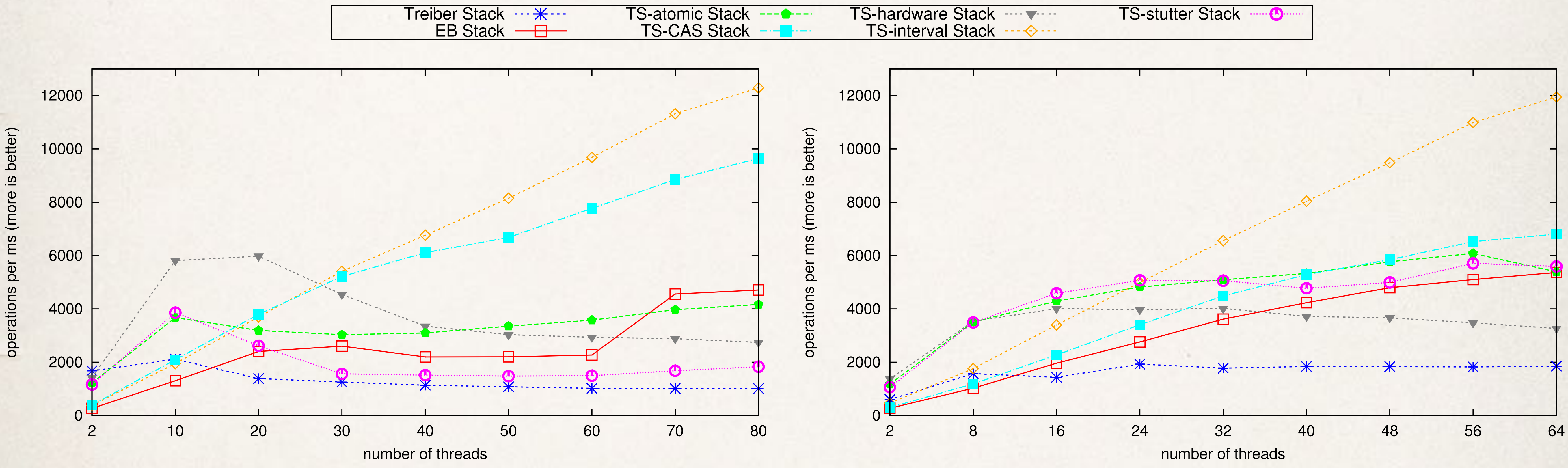
Joint Work

- Martin Aigner
- Christian Barthel
- Mike Dodds
- Andreas Haas
- Thomas Henzinger
- Andreas Holzer
- Thomas Hütter
- Michael Lippautz
- Alexander Miller
- Simone Oblasser
- Hannes Payer
- Mario Preishuber
- Ana Sokolova
- Ali Szegin

The Multicore Scalability Challenge



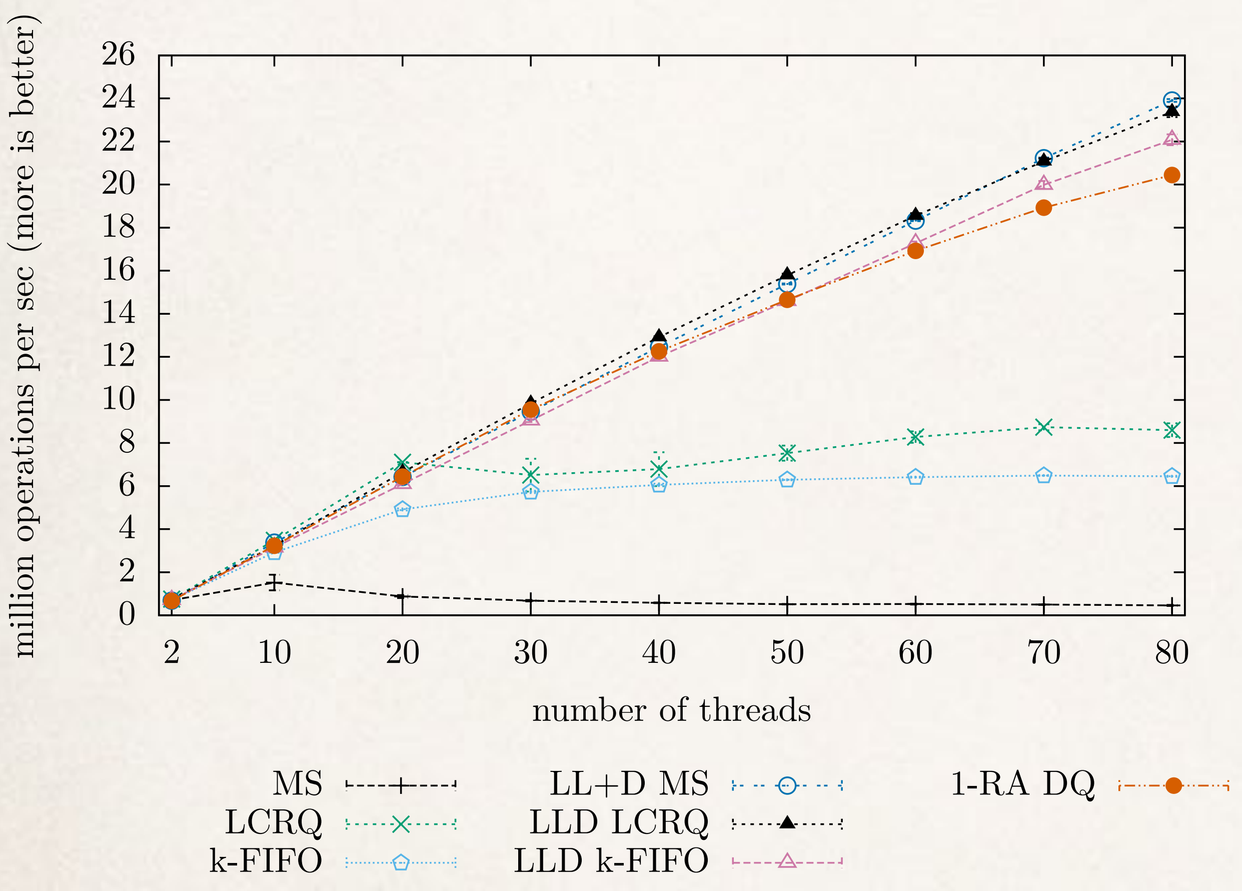
Timestamped (TS) Stack [POPL15]



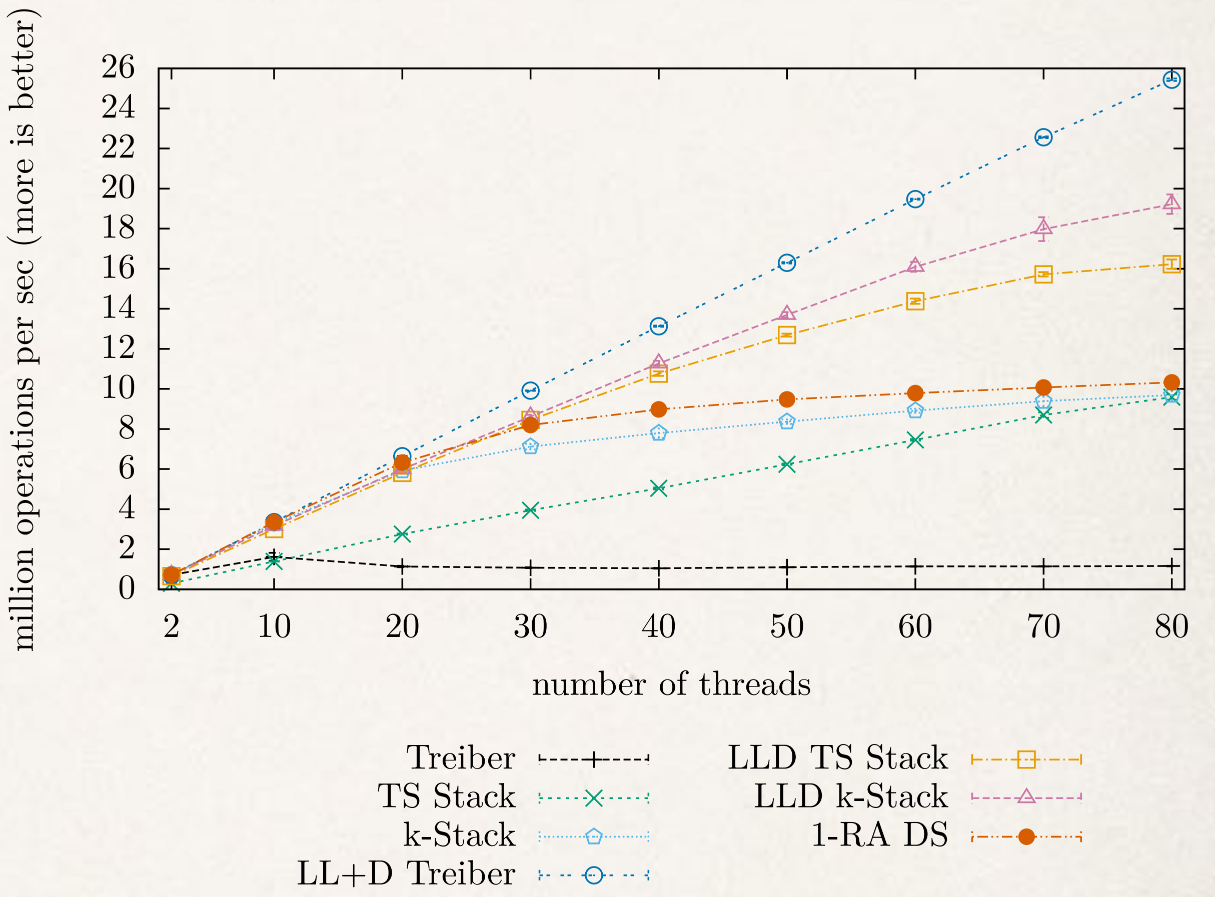
(a) Producer-consumer benchmark, 40-core machine.

(b) Producer-consumer benchmark, 64-core machine.

Local Linearizability [CONCUR16]



"queue-like" data structures



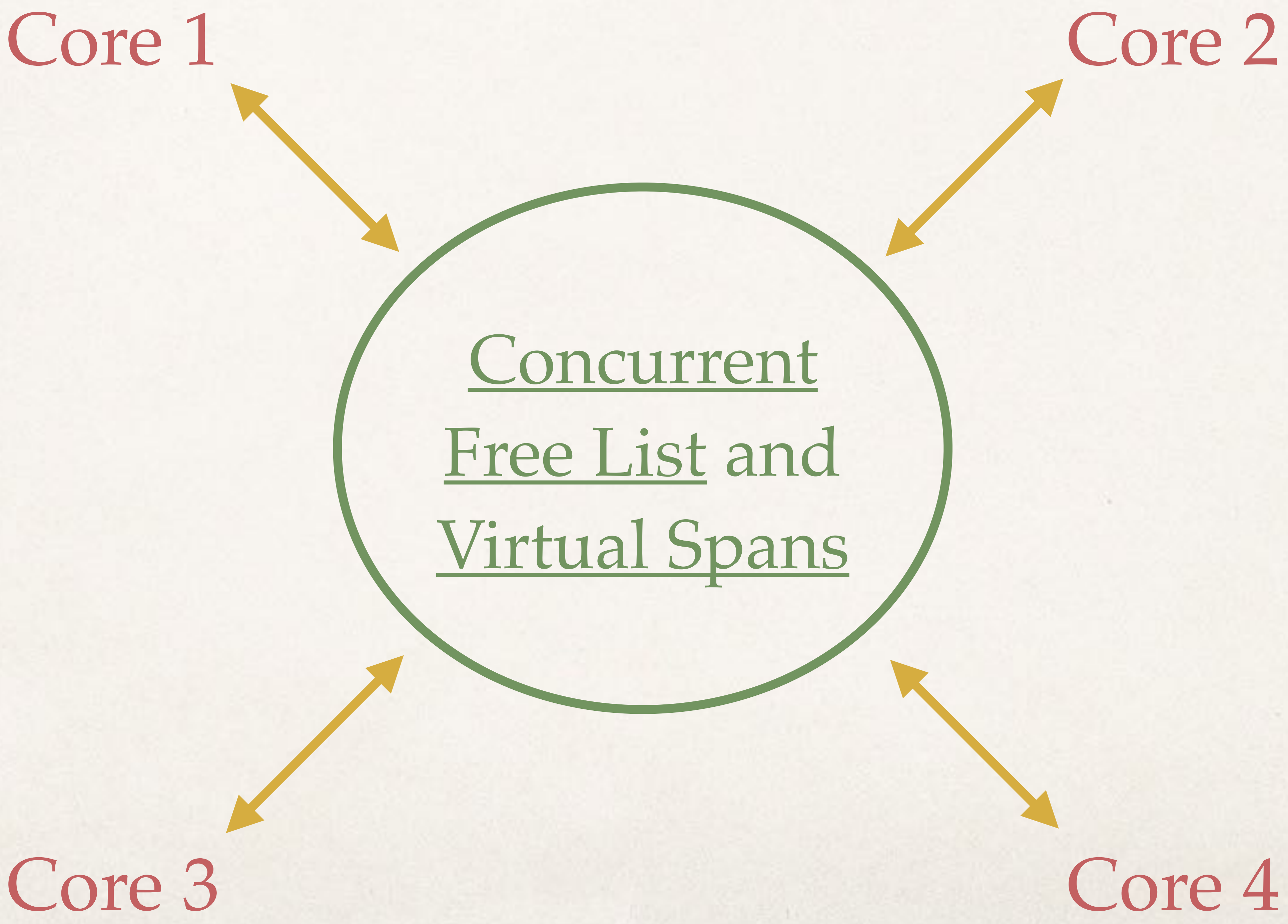
"stack-like" data structures

Figure 5 Performance and scalability of producer-consumer microbenchmarks with an increasing number of threads on a 40-core (2 hyperthreads per core) machine

Scal: A Benchmarking Suite for Concurrent Data Structures [NETYS15]

Name	Semantics	Year	Ref
Lock-based Singly-linked	strict queue	1968	[1]
Michael Scott (MS) Queue	strict queue	1996	[2]
Flat Combining Queue	strict queue	2010	[3]
Wait-free Queue	strict queue	2012	[4]
Linked Cyclic Ring Queue	strict queue	2013	[5]
Timestamped (TS) Queue	strict queue	2015	[6]
Cooperative TS Queue	strict queue	2015	[7]
Segment Queue	k-relaxed queue	2010	[8]
Random Dequeue (RD)	k-relaxed queue	2010	[8]
Bounded Size k-FIFO	k-relaxed queue, pool	2013	[9]
Unbounded Size k-FIFO	k-relaxed queue, pool	2013	[9]
b-RR Distributed Queue	k-relaxed queue, pool	2013	[10]
Least-Recently-Used (LRU)	k-relaxed queue, pool	2013	[10]
Locally Linearizable DQ	locally linearizable	2015	[11]
Locally Linearizable k-FIFO	locally linearizable	2015	[11]
Relaxed TS Queue	quiescently consistent	2015	[7]
Lock-based Singly-linked	strict stack	1968	[1]
Treiber Stack	strict stack	1986	[12]
Elimination-backoff Stack	strict stack	2004	[13]
Timestamped (TS) Stack	strict stack	2015	[6]
k-Stack	k-relaxed stack	2013	[14]
b-RR Distributed Stack (DS)	k-relaxed stack, pool	2013	[10]
Least-Recently-Used (LRU)	k-relaxed stack, pool	2013	[10]
Locally Linearizable DS	locally linearizable	2015	[11]
Locally Linearizable k-Stack	locally linearizable	2015	[11]
Timestamped (TS) Deque	strict deque	2015	[7]
d-RA DQ and DS	strict pool	2013	[10]

Scalloc: Concurrent Memory Allocator
scalloc.cs.uni-salzburg.at [OOPSLA15]



Computer Science for Everyone

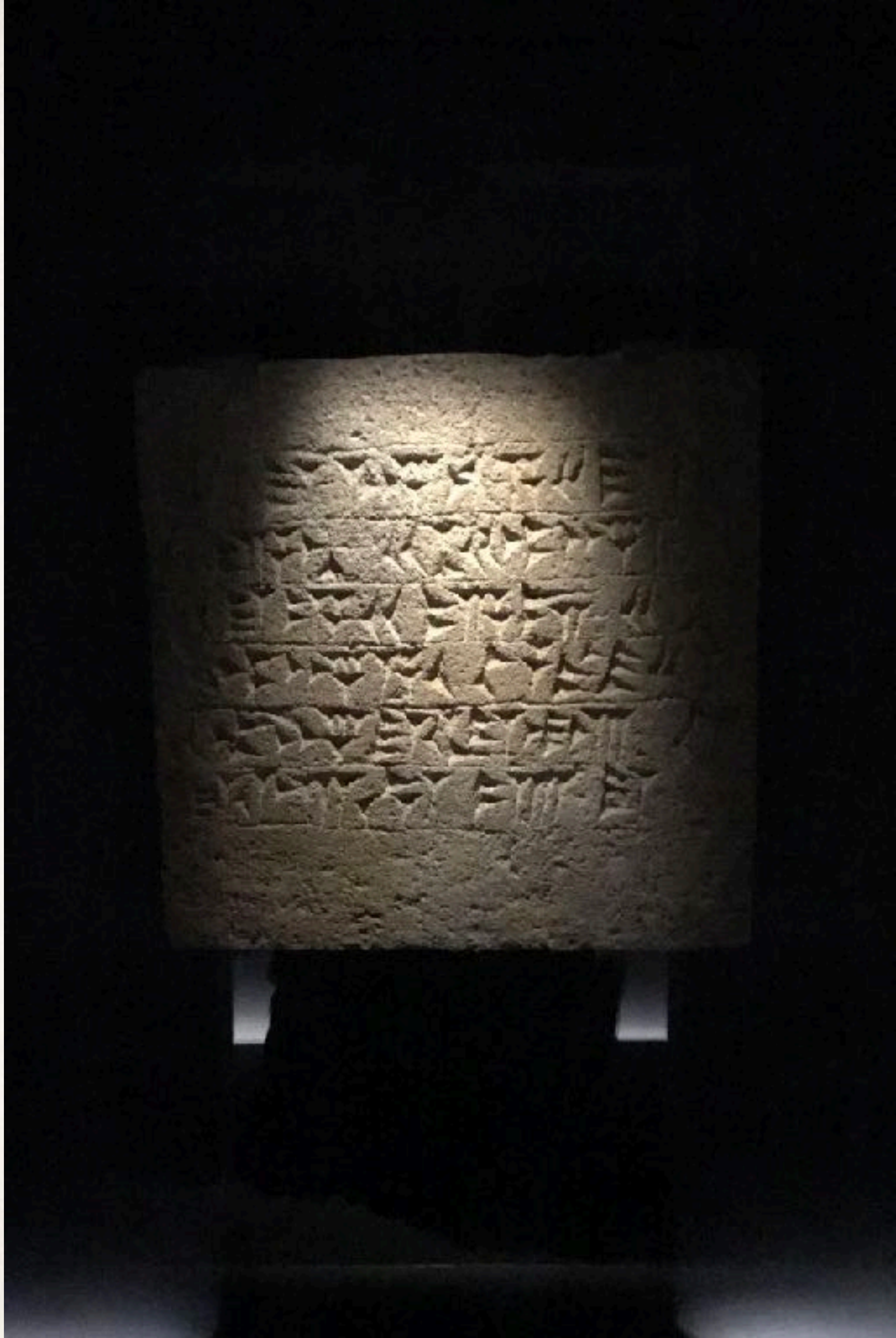
Teaching the absolute basics!

What are the absolute basics?

What is Computer Science?

To Create Meaning with a Machine

What is the meaning of this sentence?



Semantics and Self-Referentiality

Twelve Basic Principles

Where does the meaning of bits come from?

Semantics

Why is information encoded this way rather than that way?

Encoding

What is computation really?

State

How do we forget state regularly?

Regularity

How can we forget state in reverse?

Stack

How do we forget unbounded state?

Name

What is the difference between programming and computing?

Time

What is the nature of digital memory?

Memory

What is the semantics of code without running it?

Type

How do we even use an incomplete system?

Bootstrapping

What is the cost of interpretation?

Interpretation

How can we get rid of it?

Virtualization

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

    1. a self-compiling compiler called *starc* that compiles a tiny subset of C called C Star (C*) to a tiny subset of MIPS32 called MIPSter,

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

  1. a self-compiling compiler called *starc* that compiles a tiny subset of C called C Star (C*) to a tiny subset of MIPS32 called MIPSter,

  2. a self-executing emulator called *mipster* that executes MIPSter code including itself when compiled with starc,

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

  1. a self-compiling compiler called *starc* that compiles a tiny subset of C called C Star (C*) to a tiny subset of MIPS32 called MIPSter,

  2. a self-executing emulator called *mipster* that executes MIPSter code including itself when compiled with starc,

  3. a self-hosting hypervisor called *hypster* that virtualizes mipster and can host all of selfie including itself, and

Selfie: Teaching Computer Science [selfie.cs.uni-salzburg.at]

- *Selfie* is a self-referential 7k-line C implementation (in a single file) of:

  1. a self-compiling compiler called *starc* that compiles a tiny subset of C called C Star (C*) to a tiny subset of MIPS32 called MIPSter,

  2. a self-executing emulator called *mipster* that executes MIPSter code including itself when compiled with starc,

  3. a self-hosting hypervisor called *hypster* that virtualizes mipster and can host all of selfie including itself, and

  4. a tiny C* library called *libcstar* utilized by all of selfie.

Website

selfie.cs.uni-salzburg.at

Book (Draft)

leanpub.com/selfie

Code

github.com/cksystemsteaching/selfie

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures, just `int` and `int*` and dereferencing: the * operator

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures, just `int` and `int*` and dereferencing: the * operator

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures, just `int` and `int*` and dereferencing: the * operator

integer arithmetics
pointer arithmetics

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures, just `int` and `int*` and dereferencing: the * operator

character literals
string literals

integer arithmetics
pointer arithmetics

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures,
just `int` and `int*`
and dereferencing:
the * operator

character literals
string literals

integer arithmetics
pointer arithmetics

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures,
just `int` and `int*`
and dereferencing:
the * operator

character literals
string literals

integer arithmetics
pointer arithmetics

no bitwise operators
no Boolean operators

5 statements:
assignment
while
if
return
procedure()

```
int atoi(int *s) {
    int i;
    int n;
    int c;

    i = 0;
    n = 0;
    c = *(s+i);

    while (c != 0) {
        n = n * 10 + c - '0';
        if (n < 0)
            return -1;

        i = i + 1;
        c = *(s+i);
    }

    return n;
}
```

no data structures,
just `int` and `int*`
and dereferencing:
the * operator

character literals
string literals

integer arithmetics
pointer arithmetics

no bitwise operators
no Boolean operators

library: `exit`, `malloc`, `open`, `read`, `write`

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program / Machine State

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program / Machine State
4. C* / Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap
7. MIPSter Code Generator

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap
7. MIPSter Code Generator
8. Arrays versus Lists

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap
7. MIPSter Code Generator
8. Arrays versus Lists
9. Composite Data Types

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap
7. MIPSter Code Generator
8. Arrays versus Lists
9. Composite Data Types
10. MIPSter Boot Loader

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library

Compiler

Emulator

Hypervisor

`selfie.c`

1. Building Selfie
2. Encoding C* Literals
3. Program/Machine State
4. C*/Command Line Scanners
5. C* Parser and Procedures
6. Symbol Table and the Heap
7. MIPSter Code Generator
8. Arrays versus Lists
9. Composite Data Types
10. MIPSter Boot Loader
11. MIPSter Emulator

1. Semantics
2. Encoding
3. State
4. Regularity
5. Stack
6. Name
7. Time
8. Memory
9. Type
10. Bootstrapping
11. Interpretation
12. Virtualization

Selfie and the Twelve Principles

Library	1. Building Selfie	1. Semantics
	2. Encoding C* Literals	2. Encoding
	3. Program/Machine State	3. State
Compiler	4. C*/Command Line Scanners	4. Regularity
	5. C* Parser and Procedures	5. Stack
	6. Symbol Table and the Heap	6. Name
Emulator	7. MIPSter Code Generator	7. Time
	8. Arrays versus Lists	8. Memory
	9. Composite Data Types	9. Type
Hypervisor	10. MIPSter Boot Loader	10. Bootstrapping
	11. MIPSter Emulator	11. Interpretation
`selfie.c`	12. MIPSter Hypervisor	12. Virtualization

```
> make
cc -w -m32 -D'main(a,b)=main(a,char**argv)' selfie.c -o selfie
```

bootstrapping selfie using standard C compiler

```
> make
cc -w -m32 -D'main(a,b)=main(a,char**argv)' selfie.c -o selfie
```

bootstrapping selfie using standard C compiler

```
> make
cc -w -m32 -D'main(a,b)=main(a,char**argv)' selfie.c -o selfie
```

bootstrapping selfie using standard C compiler

```
> make
cc -w -m32 -D'main(a,b)=main(a,char**argv)' selfie.c -o selfie
```

bootstrapping selfie using standard C compiler

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

```
> ./selfie
./selfie: usage: selfie { -c { source } | -o binary | -s assembly
| -l binary } [ ( -m | -d | -y | -min | -mob ) size ... ]
```

selfie usage

compiling selfie with selfie (takes seconds)

```
> ./selfie -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: 176408 characters read in 7083 lines and 969 comments
./selfie: with 97779(55.55%) characters in 28914 actual symbols
./selfie: 261 global variables, 289 procedures, 450 string literals
./selfie: 1958 calls, 723 assignments, 57 while, 572 if, 243 return
./selfie: 121660 bytes generated with 28779 instructions and 6544
bytes of data
```

compiling selfie with selfie (takes seconds)

```
> ./selfie -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: 176408 characters read in 7083 lines and 969 comments
./selfie: with 97779(55.55%) characters in 28914 actual symbols
./selfie: 261 global variables, 289 procedures, 450 string literals
./selfie: 1958 calls, 723 assignments, 57 while, 572 if, 243 return
./selfie: 121660 bytes generated with 28779 instructions and 6544
bytes of data
```

compiling selfie with selfie (takes seconds)

compiling selfie with selfie <u>and</u> then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

```
> ./selfie -c selfie.c -m 2 -c selfie.c

./selfie: this is selfie's starc compiling selfie.c

./selfie: this is selfie's mipster executing selfie.c with 2MB of
physical memory

selfie.c: this is selfie's starc compiling selfie.c

selfie.c: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie.c with exit code
0 and 1.16MB of mapped memory
```

compiling selfie with selfie and then running that executable to compile selfie again (takes ~6 minutes)

*compiling selfie with selfie <u>and</u> generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

compiling selfie with selfie and generating an executable ***selfie1.m*** *that is then executed to compile selfie again generating another executable* ***selfie2.m*** *(takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

```
> ./selfie -c selfie.c -o selfie1.m -m 2 -c selfie.c -o selfie2.m

./selfie: this is selfie's starc compiling selfie.c
./selfie: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie1.m

./selfie: this is selfie's mipster executing selfie1.m with 2MB of
physical memory

selfie1.m: this is selfie's starc compiling selfie.c
selfie1.m: 121660 bytes with 28779 instructions and 6544 bytes of data
written into selfie2.m

selfie1.m: exiting with exit code 0 and 1.05MB of mallocated memory

./selfie: this is selfie's mipster terminating selfie1.m with exit
code 0 and 1.16MB of mapped memory
```

*compiling selfie with selfie and generating an executable **selfie1.m** that is then executed to compile selfie again generating another executable **selfie2.m** (takes ~6 minutes)*

*compiling selfie with selfie <u>and</u> then running that executable to compile selfie again <u>and</u> then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie <u>and</u> then running that executable to compile selfie again <u>and</u> then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie and then running that executable to compile selfie again and then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie <u>and</u> then running that executable to compile selfie again <u>and</u> then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie and then running that executable to compile selfie again and then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie and then running that executable to compile selfie again and then running that executable to compile selfie again (**takes ~24 hours**)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -m 2 -c selfie.c
```

*compiling selfie with selfie <u>and</u> then running that executable to compile selfie again <u>and</u> then running that executable to compile selfie again (**takes ~24 hours**)*

compiling selfie with selfie and then running that executable to compile selfie again and then ***hosting*** *that executable in a virtual machine to compile selfie again (****takes ~12 minutes****)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -y 2 -c selfie.c
```

compiling selfie with selfie <u>and</u> then running that executable to compile selfie again <u>and</u> then ***hosting*** *that executable in a virtual machine to compile selfie again (****takes ~12 minutes****)*

```
> ./selfie -c selfie.c -m 2 -c selfie.c -y 2 -c selfie.c
```

compiling selfie with selfie and then running that executable to compile selfie again and then ***hosting*** *that executable in a virtual machine to compile selfie again (****takes ~12 minutes****)*

Thank you!