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

Helmut Veith Memorial Workshop, Obergurgl, February 2017

Joint Work

- Martin Aigner, teaching assistant
- Sebastian Arming, teaching assistant
- Christian Barthel, bachelor thesis
  RISC-V port, presented @ Google PhD Summit
- Michael Lippautz, original emulator design
- Simone Oblasser, bachelor thesis
  RISC-V port, presented @ Google PhD Summit

Inspiration

- Niklaus Wirth: Compiler Construction
- Jochen Liedtke: Microkernels

Computer Science for Everyone

nsf.gov / csforall

code.org

computingatschool.org.uk

programbydesign.org

bootstrapworld.org

k12cs.org

csfieldguide.org.nz

Teaching the absolute basics!

What are the absolute basics?

What is Computer Science?

To Create Meaning with a Machine

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,

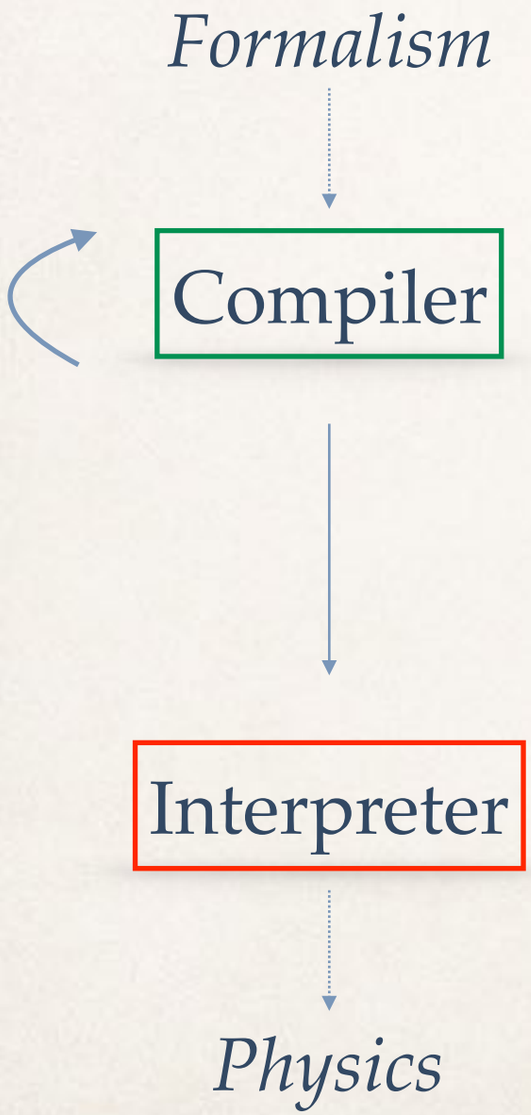
  2. a self-executing emulator called *mipster* that executes MIPSter code including itself when compiled with starc,

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

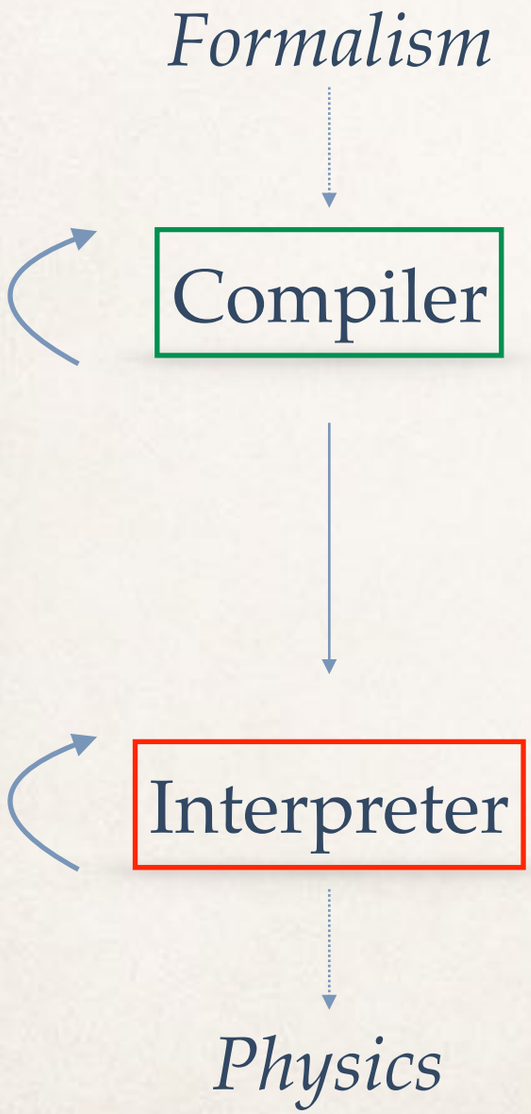
  4. a tiny C* library called *libcstar* utilized by all of selfie.

Semantics and Performance

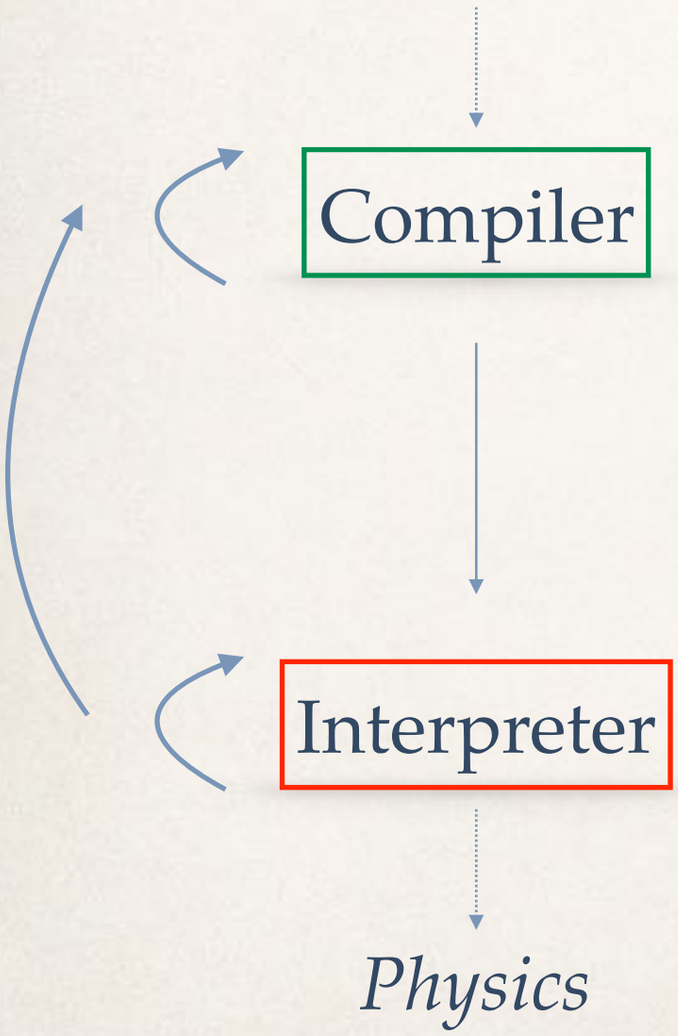
Semantics and Performance



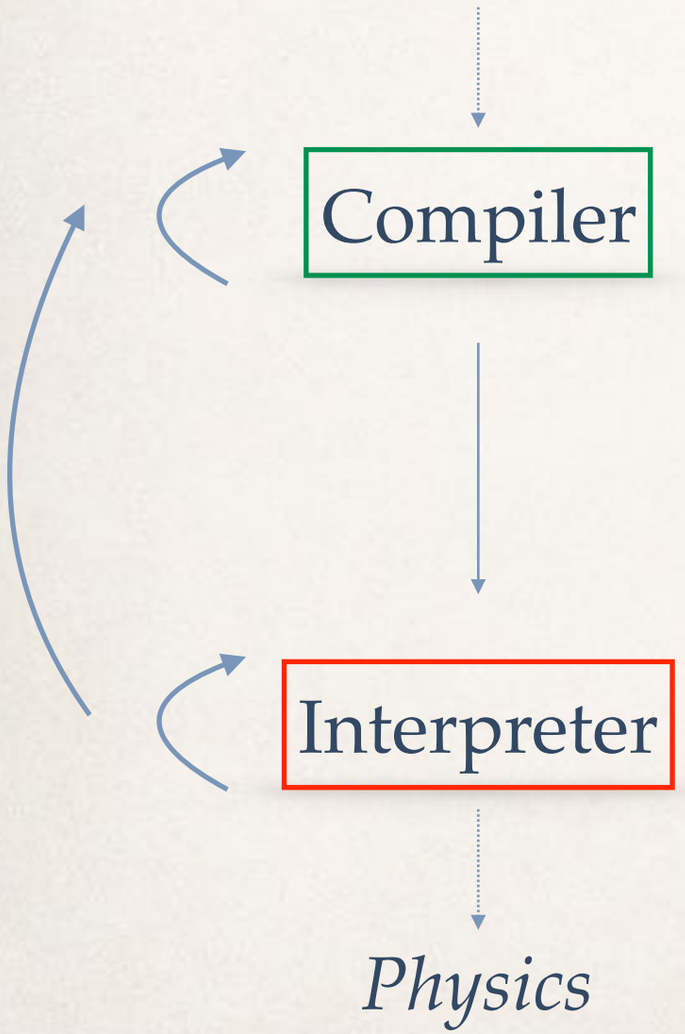
Semantics and Performance

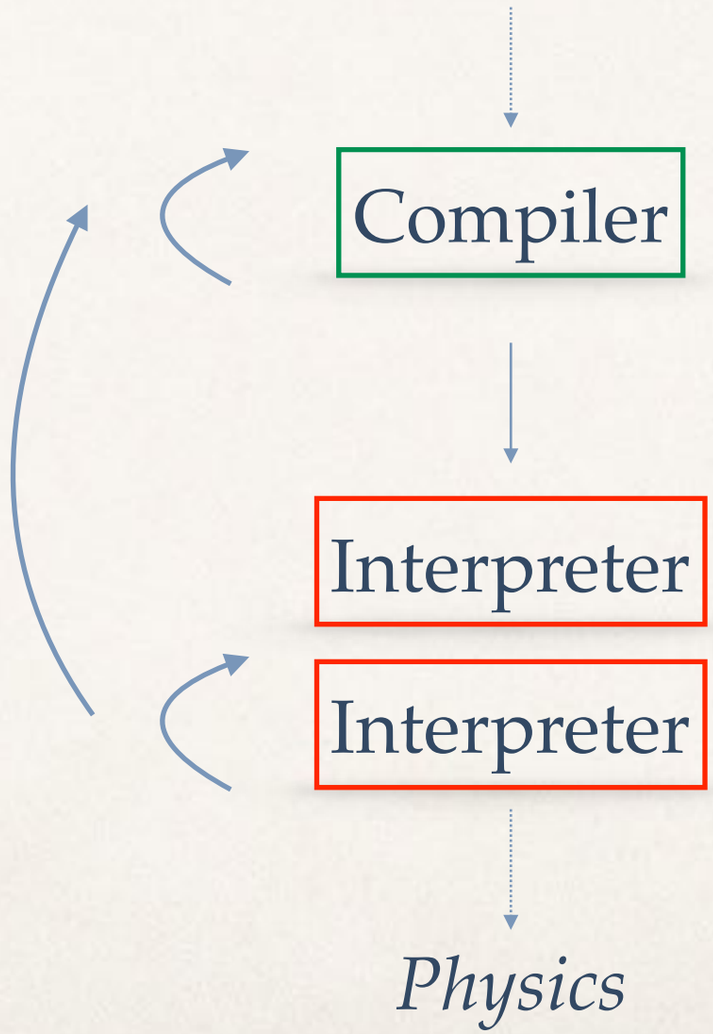


Semantics and Performance

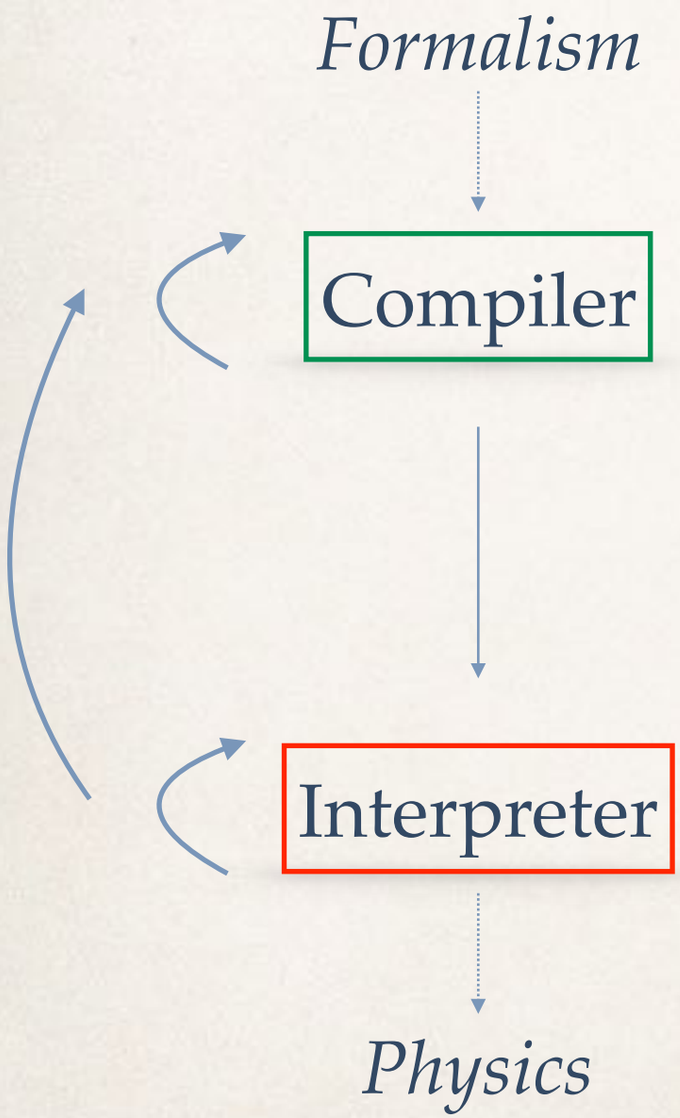


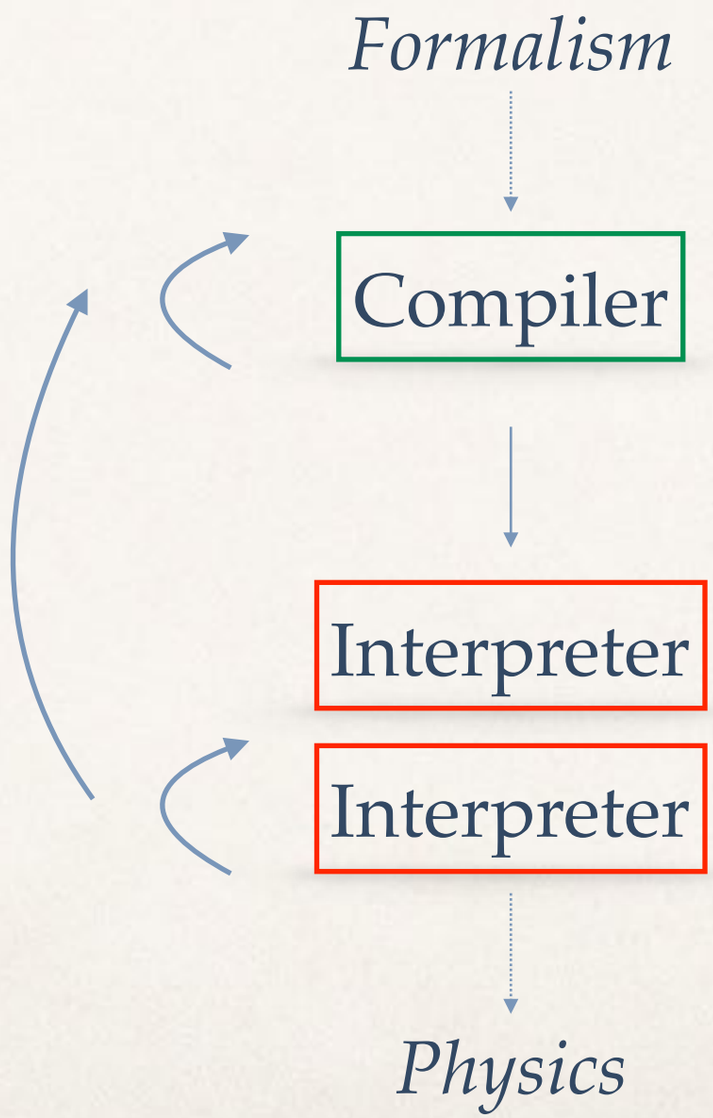
Semantics and Performance

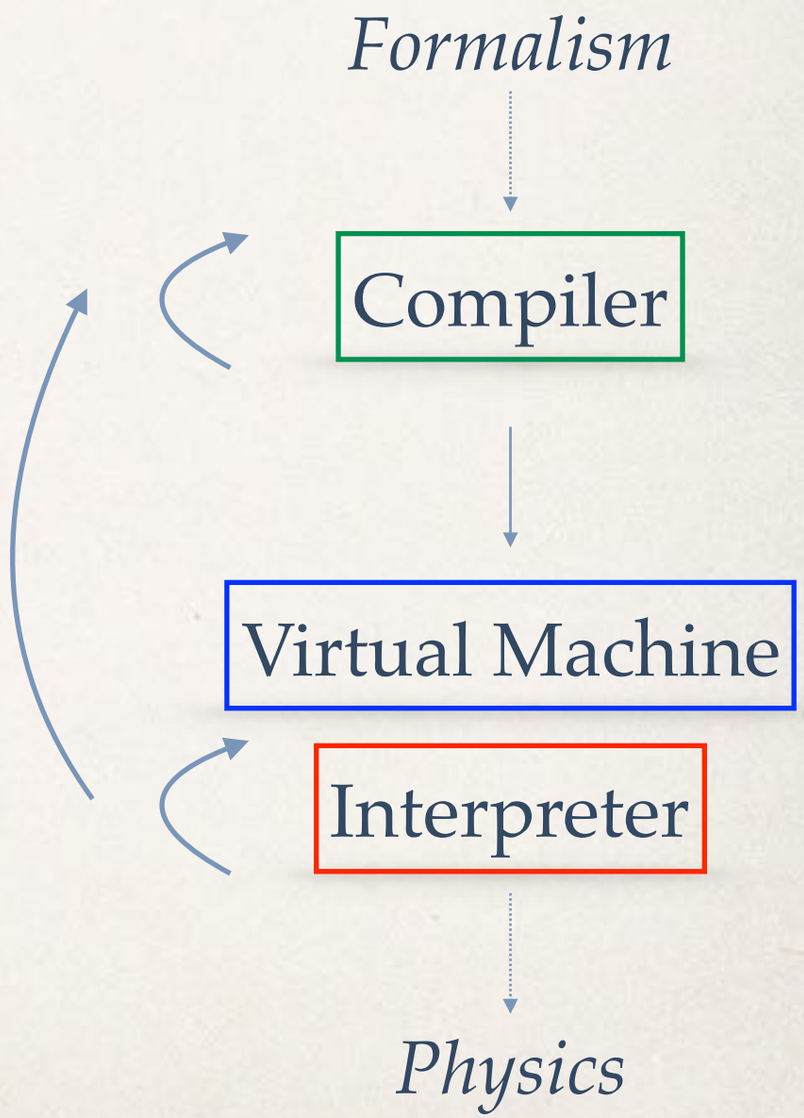




Semantics and Performance







```
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;
}
```

```
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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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 Twelve Basic 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. Memory Management

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 Twelve Basic 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. Memory Management
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 Twelve Basic 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. Memory Management
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 Twelve Basic 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. Memory Management
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 Twelve Basic 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. Memory Management
9. Composite Data Types
10. MIPSter Boot Loader
11. MIPSter Emulator
12. MIPSter Hypervisor

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

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

bootstrapping `selfie.c` *into x86* `selfie` *executable*
using standard C compiler

(now also available for RISC-V machines)

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

bootstrapping `selfie.c` *into x86* `selfie` *executable*
using standard C compiler

(now also available for RISC-V machines)

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

bootstrapping `selfie.c` *into x86* `selfie` *executable*
using standard C compiler

(now also available for RISC-V machines)

```
> ./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.c` *with x86* `selfie` *executable*

(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.c` *with x86* `selfie` *executable*

(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.c` *with x86* `selfie` *executable*

(takes seconds)

compiling `selfie.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *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.c` *with x86* `selfie` *executable into a MIPSter executable*

and

then running that MIPSter executable to compile `selfie.c` *again*

(takes ~6 minutes)

compiling `selfie.c` *into a MIPSter executable* `selfie1.m`

and

then running `selfie1.m` *to compile* `selfie.c`

into another MIPSter 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.c` *into a MIPSter executable* `selfie1.m`

and

then running `selfie1.m` *to compile* `selfie.c`

into another MIPSter 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.c` *into a MIPSter executable* `selfie1.m`

and

then running `selfie1.m` *to compile* `selfie.c`

into another MIPSter 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.c` *into a MIPSter executable* `selfie1.m`

and

then running `selfie1.m` *to compile* `selfie.c`

into another MIPSter executable `selfie2.m`

(takes ~6 minutes)

compiling `selfie.c` *with x86* `selfie` executable

and

then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

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

compiling `selfie.c` *with x86* `selfie` executable

and

then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

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

compiling `selfie.c` *with x86* `selfie` executable

and

then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

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

compiling `selfie.c` *with x86* `selfie` executable

and

then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

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

compiling `selfie.c` *with x86* `selfie` executable

and

then running that executable to compile `selfie.c` *again*

and

then running that executable to compile `selfie.c` *again*

(takes ~24 hours)

"The OS is an interpreter until people wanted speed."

–selfie

compiling `selfie.c` *with x86* `selfie` *executable*

and

then running that executable to compile `selfie.c` *again*

and

then ***hosting*** *that executable in a virtual machine to compile* `selfie.c` *again*

(takes ~12 minutes)

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

compiling `selfie.c` *with x86* `selfie` *executable*

and

then running that executable to compile `selfie.c` *again*

and

then ***hosting*** *that executable in a virtual machine to compile* `selfie.c` *again*

(takes ~12 minutes)

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

compiling `selfie.c` *with x86* `selfie` *executable*

and

then running that executable to compile `selfie.c` *again*

and

then ***hosting*** *that executable in a virtual machine to compile* `selfie.c` *again*

(takes ~12 minutes)

Website

selfie.cs.uni-salzburg.at

Book (Draft)

leanpub.com/selfie

Code

github.com/cksystemsteaching/selfie

Thank you!