Take a Selfie in Class

Christoph M. Kirsch

SPLASH-E 2018 @ SPLASH, Boston, Massachusetts, November 2018

Teaching versus Research: What is more important?

Research is a side-effect of teaching

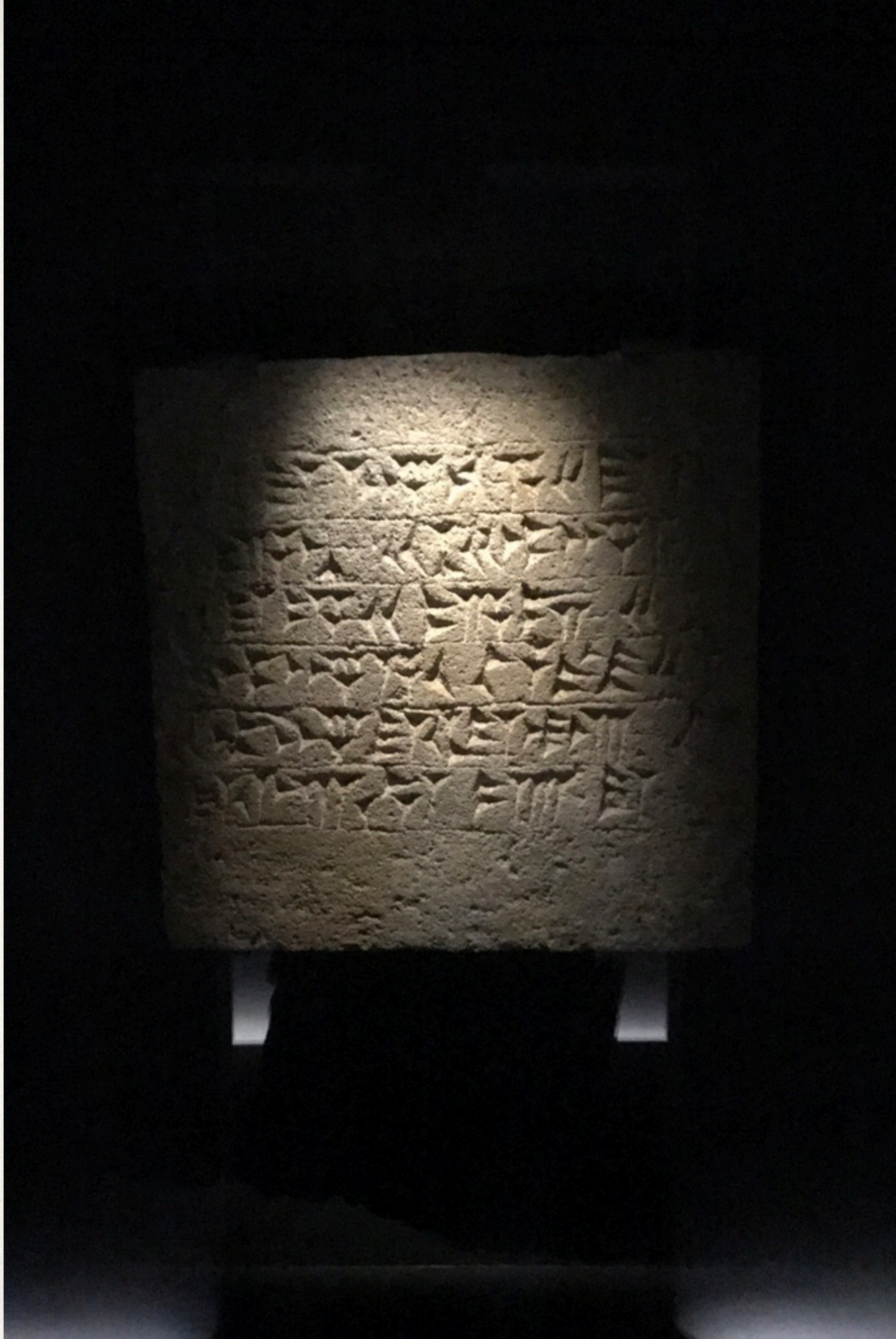
Producing just two students that are better than you may be enough!

How would broadly acknowledging that change funding and science in general?

selfie.cs.uni-salzburg.at

What is the meaning of this sentence?

Selfie as in self-referentiality



Interpretation

Compilation

Teaching the Construction of Semantics of Formalisms

Virtualization

Verification

Joint Work

- Alireza Abyaneh
- Martin Aigner
- Sebastian Arming
- Christian Barthel
- Simon Bauer
- Thomas Hütter
- Alexander Kollert
- Michael Lippautz
- Cornelia Mayer
- Philipp Mayer
- Christian Moesl
- Simone Oblasser
- Clement Poncelet
- Sara Seidl
- Ana Sokolova
- Manuel Widmoser

Inspiration

- Armin Biere: SAT/SMT Solvers
- Donald Knuth: Art
- Jochen Liedtke: Microkernels
- Hennessy/Patterson: RISC
- Niklaus Wirth: Compilers

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

- *Selfie* is a self-referential 10k-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 RISC-V called RISC-U,

2. a self-executing emulator called ***mipster*** that executes RISC-U 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,

4. a self-executing symbolic execution engine called ***monster*** that executes RISC-U code symbolically when compiled with starc which includes all of selfie,

5. a tiny C* library called ***libcstar*** utilized by all of selfie, and

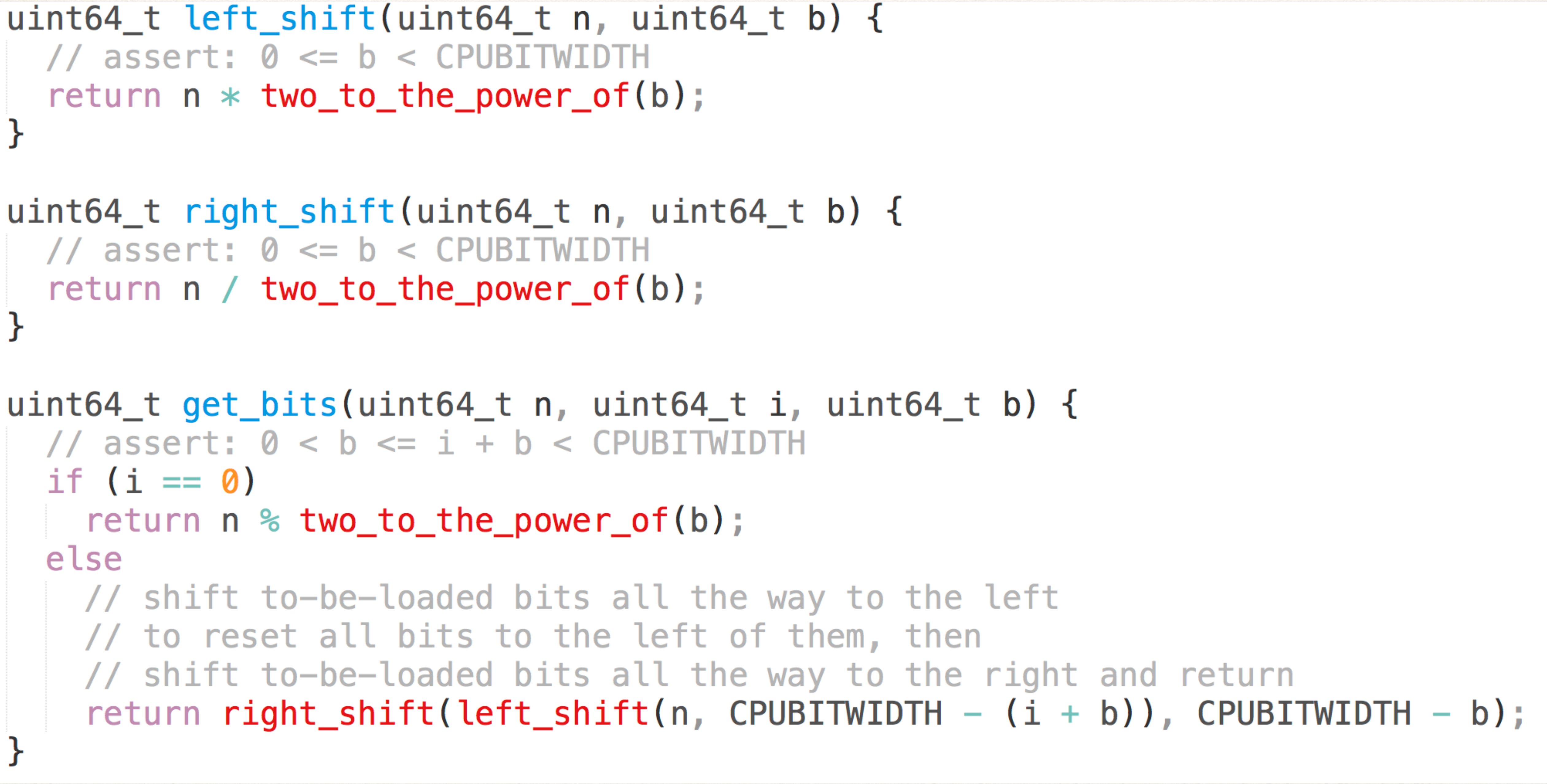
6. a tiny, experimental SAT solver called ***babysat***.

Selfie supports the official 64-bit RISC-V toolchain and runs on the spike emulator and the pk kernel

Also, there is a…

- linker (in-memory only)
- disassembler (w/ source code line numbers)
- debugger (tracks full machine state w/ rollback)
- profiler (#proc-calls, #loop-iterations, #loads, #stores)
- ELF boot loader (same code for mipster/hypster)

Code as Prose



```
uint64_t left_shift(uint64_t n, uint64_t b) {
  // assert: 0 <= b < CPUBITWIDTH
  return n * two_to_the_power_of(b);
}

uint64_t right_shift(uint64_t n, uint64_t b) {
  // assert: 0 <= b < CPUBITWIDTH
  return n / two_to_the_power_of(b);
}

uint64_t get_bits(uint64_t n, uint64_t i, uint64_t b) {
  // assert: 0 < b <= i + b < CPUBITWIDTH
  if (i == 0)
    return n % two_to_the_power_of(b);
  else
    // shift to-be-loaded bits all the way to the left
    // to reset all bits to the left of them, then
    // shift to-be-loaded bits all the way to the right and return
    return right_shift(left_shift(n, CPUBITWIDTH - (i + b)), CPUBITWIDTH - b);
}
```

Discussion of Selfie reached 3rd place on Hacker News

news.ycombinator.com

Website

selfie.cs.uni-salzburg.at

Code

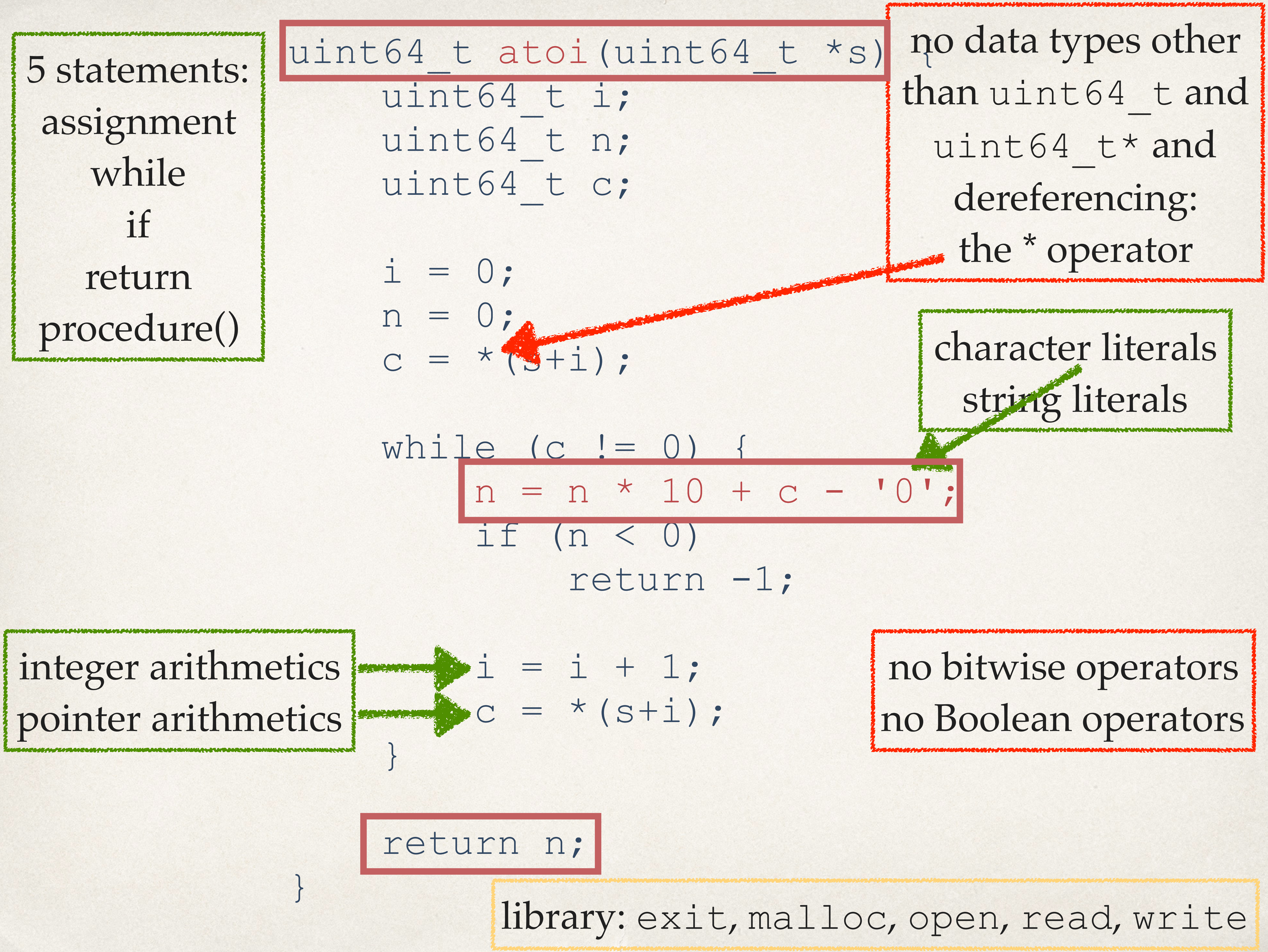
github.com/cksystemsteaching/selfie

Slides (400 done, ~100 todo)

selfie.cs.uni-salzburg.at/slides

Book (draft)

leanpub.com/selfie



5 statements:
assignment
while
if
return
procedure()

```
uint64_t atoi(uint64_t *s) {
    uint64_t i;
    uint64_t n;
    uint64_t 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 types other than `uint64_t` and `uint64_t*` and dereferencing: the * operator

character literals
string literals

integer arithmetics
pointer arithmetics

no bitwise operators
no Boolean operators

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

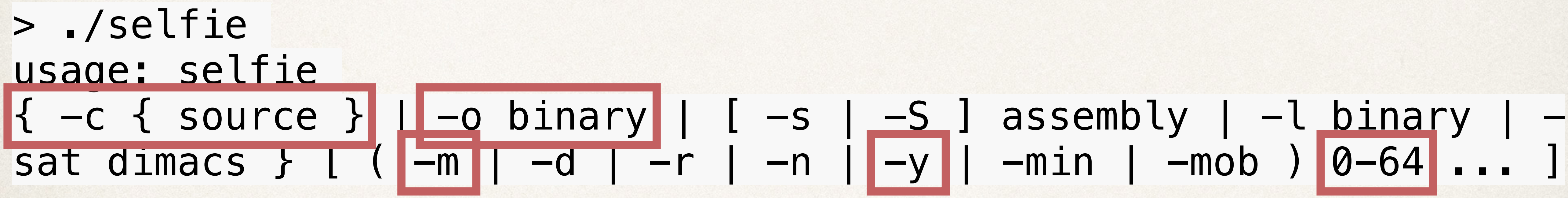
Minimally complex, maximally self-contained system

Programming languages vs systems engineering?

```
> make
cc -w -O3 -m64 -D'main(a,b)=main(int argc, char** argv)' \
-Duint64_t=‘unsigned long long' selfie.c -o selfie
```

bootstrapping `selfie.c` *into x86* `selfie` *executable*

using standard C compiler



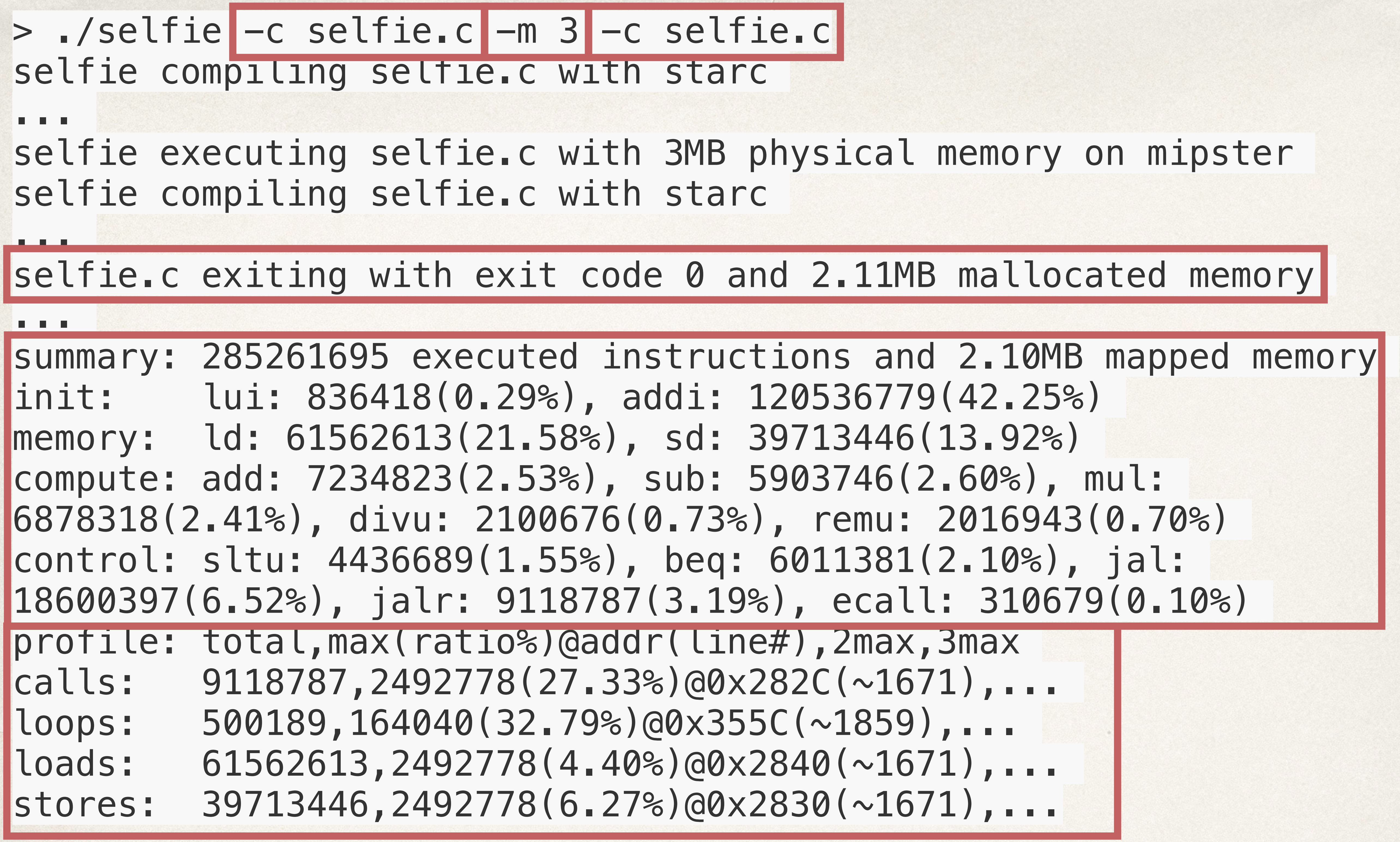
```
> ./selfie
usage: selfie
{ -c { source } | -o binary | [ -s | -S ] assembly | -l binary | -
sat dimacs } [ ( -m | -d | -r | -n | -y | -min | -mob ) 0-64 ... ]
```

selfie usage

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

selfie compiling selfie.c with starc

289095 characters read in 10034 lines and 1335 comments
with 170555(58.99%) characters in 43772 actual symbols
341 global variables, 438 procedures, 411 string literals
2517 calls, 1139 assignments, 86 while, 874 if, 391 return
symbol table search time was 2 iterations on average and
48795 in total

170504 bytes generated with 39496 instructions and 12520 bytes of data

init:    lui: 2296(5.81%), addi: 13595(34.40%)
memory:  ld: 7106(17.98%), sd: 5884(14.89%)
compute: add: 3422(8.65%), sub: 704(1.78%), mul: 807(2.40%),
         divu: 78(0.19%), remu: 35(0.80%)
control: sltu: 624(1.57%), beq: 964(2.43%),
         jal: 3555(8.99%), jalr: 438(1.10%), ecall: 8(0.20%)
```

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

(takes seconds)



```
> ./selfie -c selfie.c -m 3 -c selfie.c
selfie compiling selfie.c with starc
...
selfie executing selfie.c with 3MB physical memory on mipster
selfie compiling selfie.c with starc
...
selfie.c exiting with exit code 0 and 2.11MB mallocated memory
...
summary: 285261695 executed instructions and 2.10MB mapped memory
init:    lui: 836418(0.29%), addi: 120536779(42.25%)
memory:  ld: 61562613(21.58%), sd: 39713446(13.92%)
compute: add: 7234823(2.53%), sub: 5903746(2.60%), mul:
6878318(2.41%), divu: 2100676(0.73%), remu: 2016943(0.70%)
control: sltu: 4436689(1.55%), beq: 6011381(2.10%), jal:
18600397(6.52%), jalr: 9118787(3.19%), ecall: 310679(0.10%)
profile: total,max(ratio%)@addr(line#),2max,3max
calls:   9118787,2492778(27.33%)@0x282C(~1671),...
loops:   500189,164040(32.79%)@0x355C(~1859),...
loads:   61562613,2492778(4.40%)@0x2840(~1671),...
stores:  39713446,2492778(6.27%)@0x2830(~1671),...
```

compiling `selfie.c` *with x86* `selfie` *executable into a RISC-U executable*

and

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

(takes a minute)

```
> ./selfie -c selfie.c -o selfie1.m -m 3 -c selfie.c -o selfie2.m

selfie compiling selfie.c with starc
...
170632 bytes with 39496 instructions and 12520 bytes of data written
into selfie1.m

selfie executing selfie1.m with 3MB physical memory on mipster
selfie compiling selfie.c with starc

...
170632 bytes with 39496 instructions and 12520 bytes of data written
into selfie2.m

selfie1.m exiting with exit code 0 and 2.11MB mallocated memory
...
summary: 285338515 executed instructions and 2.10MB mapped memory
```

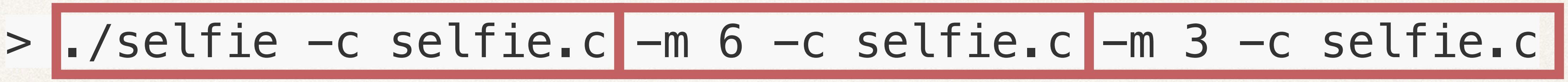
compiling `selfie.c` *into a RISC-U executable* `selfie1.m`

and

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

into another RISC-U executable `selfie2.m`

(takes a minute)



```
> ./selfie -c selfie.c -m 6 -c selfie.c -m 3 -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 hours)

```
> ./selfie -c selfie.c -m 6 -c selfie.c -y 3 -c selfie.c
```

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

and

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

and

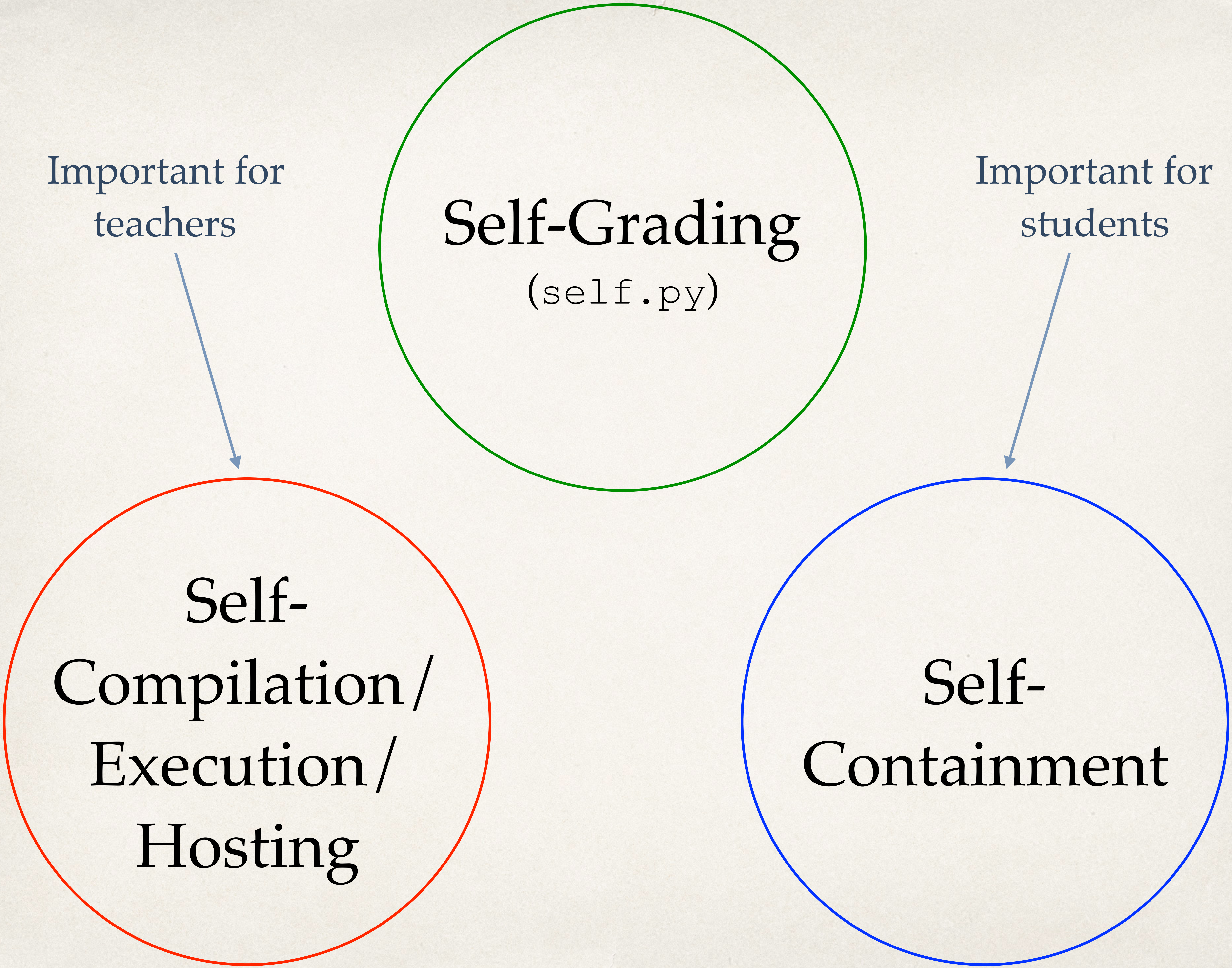
then ***hosting*** *that executable in a virtual machine to compile* `selfie.c` *again*

*(**takes 2 minutes**)*

Take a Selfie in Class

How can we leverage self-referentiality in teaching?

Self-Grading :-)



Important for teachers

Self-Grading

(`self.py`)

Important for students

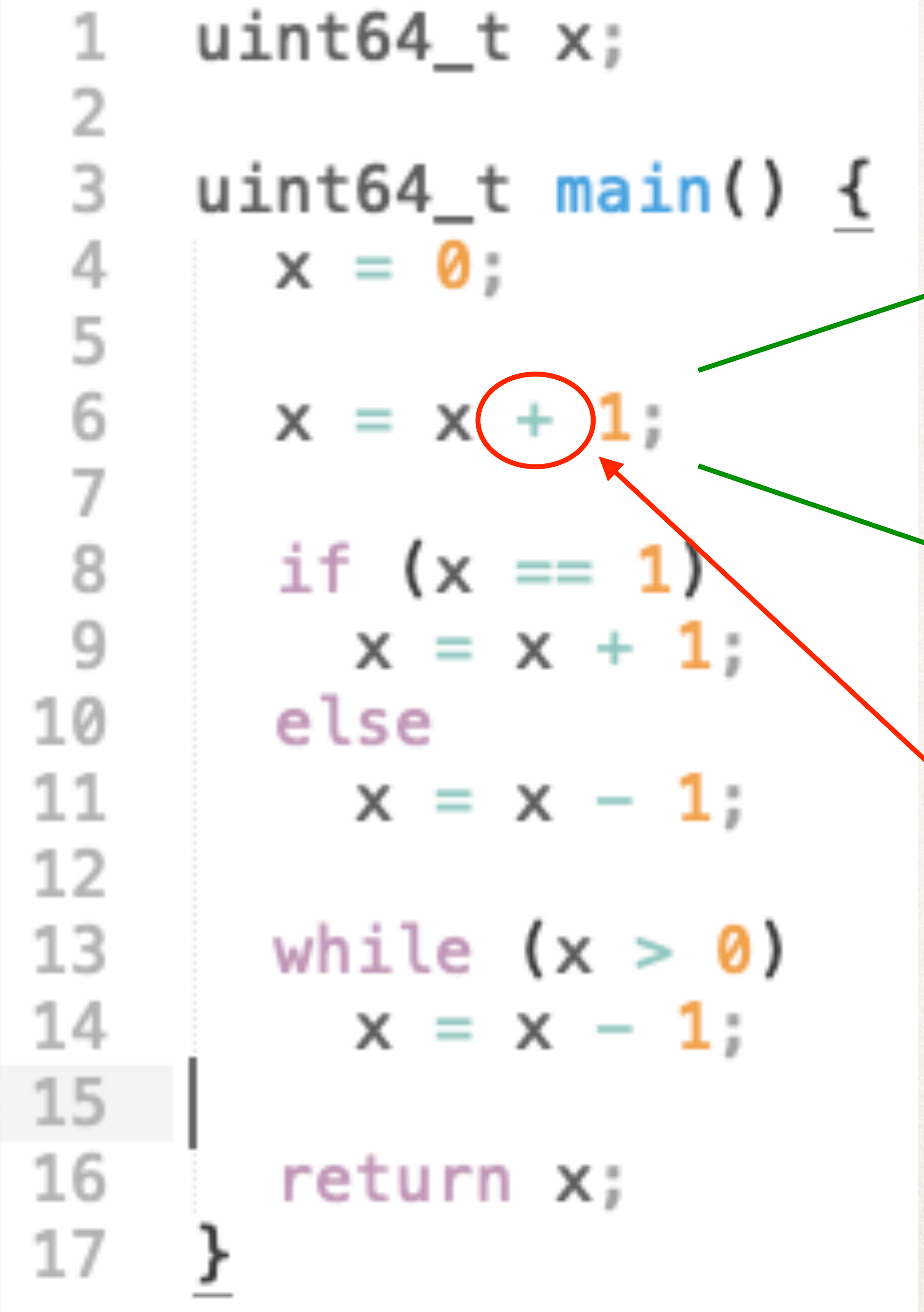
Self-Compilation/Execution/Hosting

Self-Containment

C*	RISC-U
`<<  >>`	`sll  srl`
~~left_shift()~~ ~~right_shift()~~	syscalls
compiler disassembler	emulator hypervisor

profiler / debugger

unsigned + code



```
uint64_t x;

uint64_t main() {
  x = 0;

  x = x + 1;

  if (x == 1)
    x = x + 1;
  else
    x = x - 1;

  while (x > 0)
    x = x - 1;

  return x;
}
```

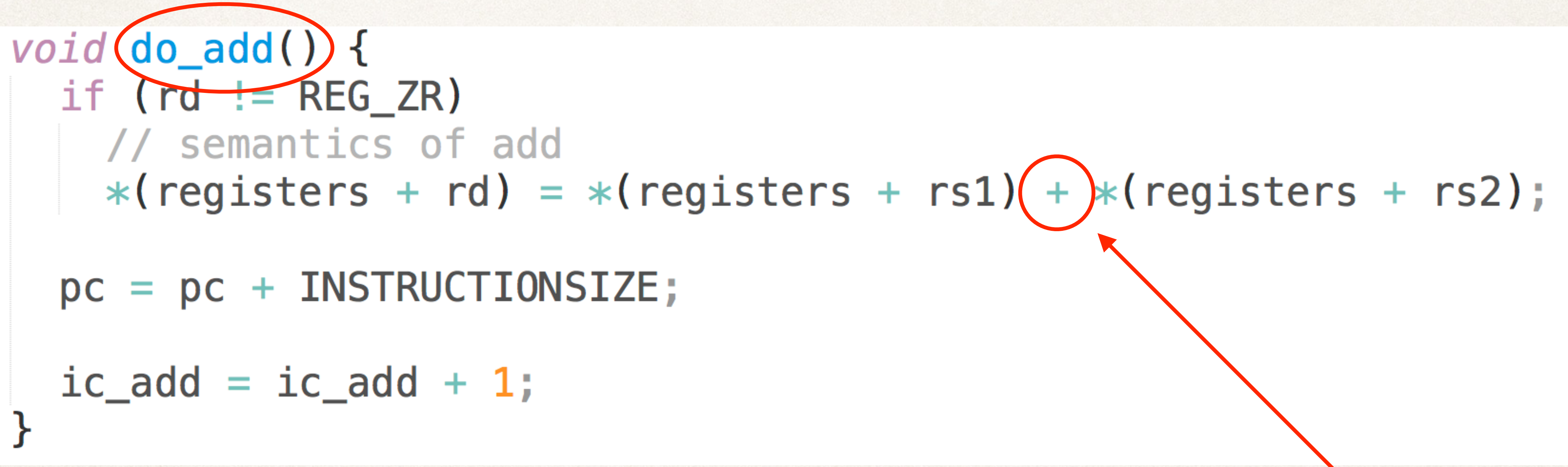
64-bit RISC-V add instruction

```
0x150(~6): ld $t0,-16($gp)
0x154(~6): addi $t1,$zero,1
0x158(~6): add $t0,$t0,$t1
0x15C(~6): sd $t0,-16($gp)
```

C code for unsigned 64-bit integer addition

unsigned + and add

64-bit RISC-V add instruction



```
void do_add() {
  if (rd != REG_ZR)
    // semantics of add
    *(registers + rd) = *(registers + rs1) + *(registers + rs2);

  pc = pc + INSTRUCTIONSIZE;

  ic_add = ic_add + 1;
}
```

C code for unsigned 64-bit integer addition

selfie compiler

gcc/clang

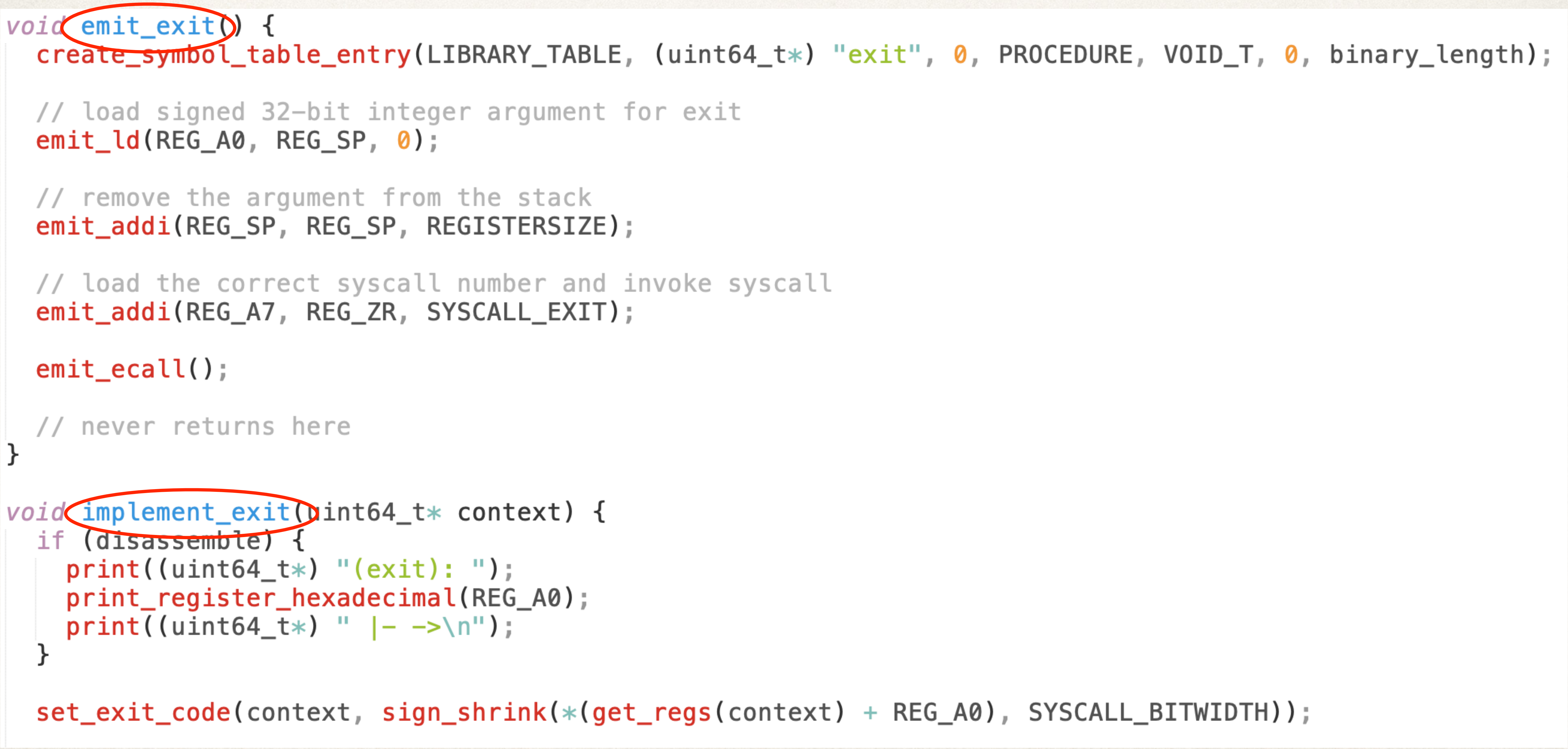
Language Homework Ideas

- Implement bitwise operators such as bitwise shifting (<<, >> as well as `sll`, `srl`)
- Multi-dimensional arrays and recursive structs
- Characters, signed integers, `sizeof()`
- Lazy evaluation of Boolean operators

C*	RISC-U
fork()	ecall fork
compiler disassembler	emulator hypervisor

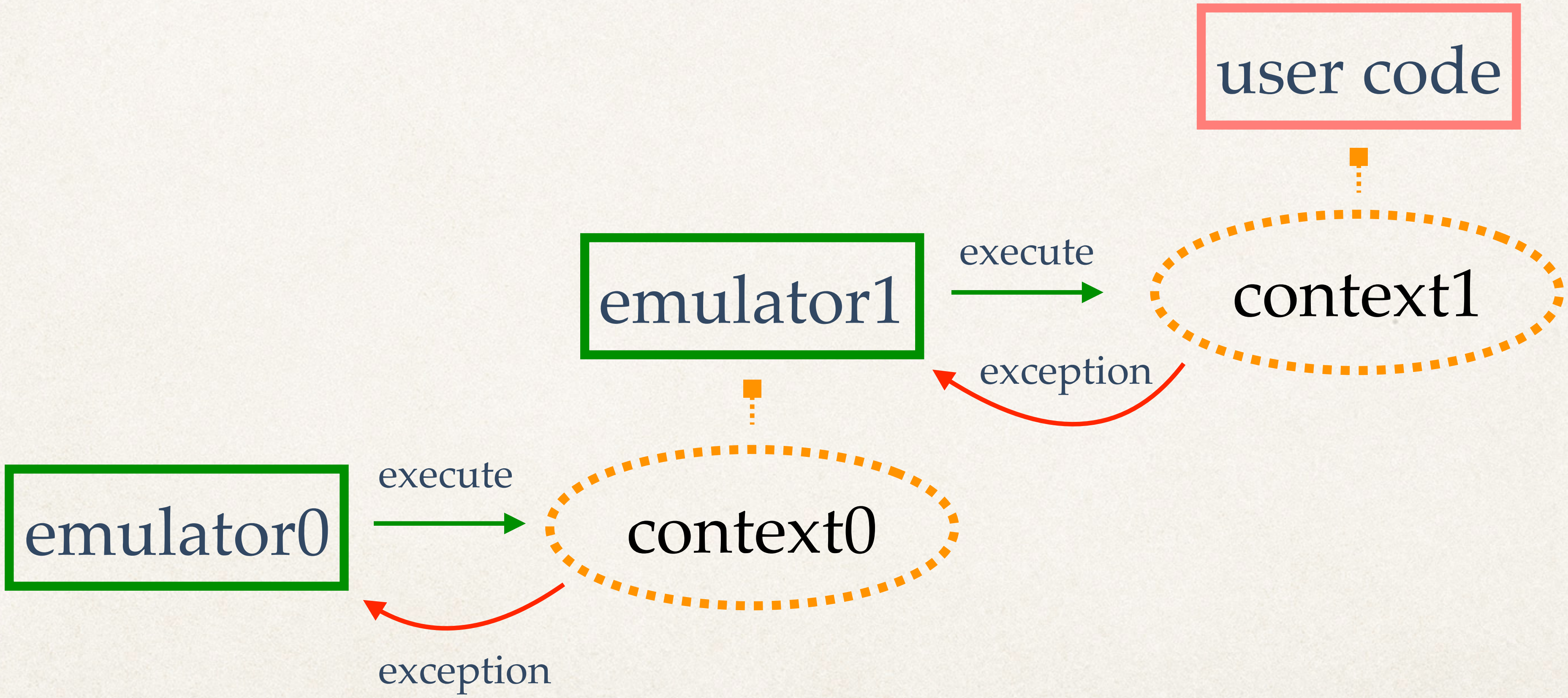
profiler / debugger

Synergy of Compiler & Emulator & Hypervisor

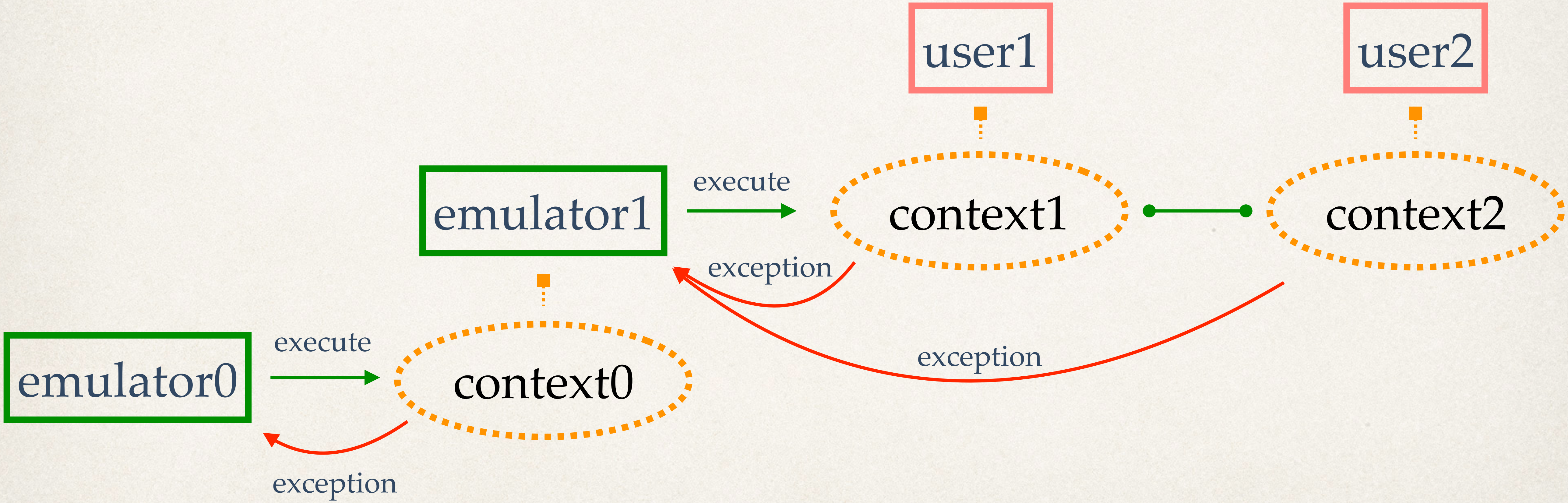


```
void emit_exit() {
  create_symbol_table_entry(LIBRARY_TABLE, (uint64_t*) "exit", 0, PROCEDURE, VOID_T, 0, binary_length);

  // load signed 32-bit integer argument for exit
  emit_ld(REG_A0, REG_SP, 0);

  // remove the argument from the stack
  emit_addi(REG_SP, REG_SP, REGISTERSIZE);

  // load the correct syscall number and invoke syscall
  emit_addi(REG_A7, REG_ZR, SYSCALL_EXIT);

  emit_ecall();

  // never returns here
}

void implement_exit(uint64_t* context) {
  if (disassemble) {
    print((uint64_t*) "(exit): ");
    print_register_hexadecimal(REG_A0);
    print((uint64_t*) " |- ->\n");
  }

  set_exit_code(context, sign_shrink(*(get_regs(context) + REG_A0), SYSCALL_BITWIDTH));
```

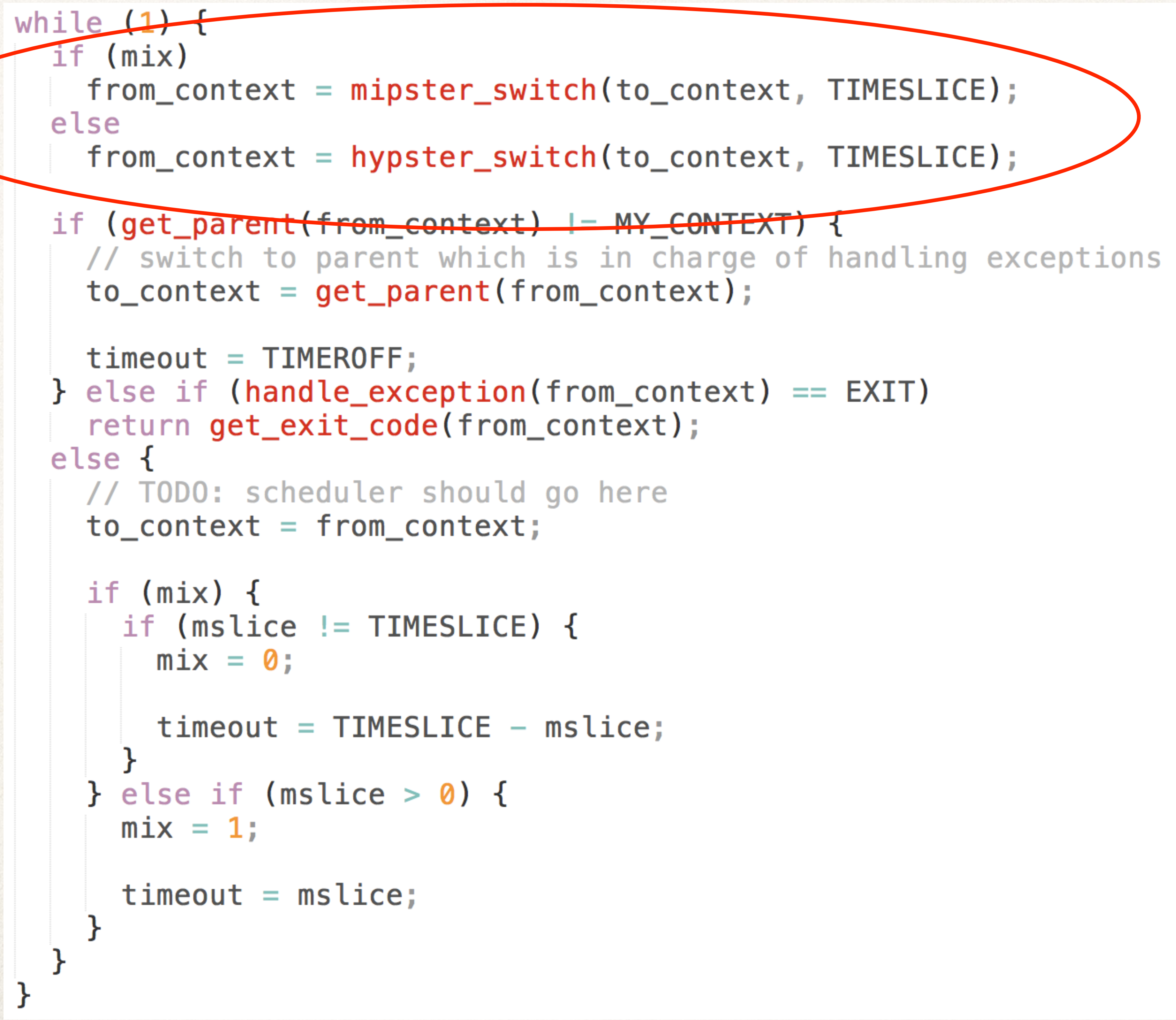
Self-Execution



Self-Execution: Concurrency

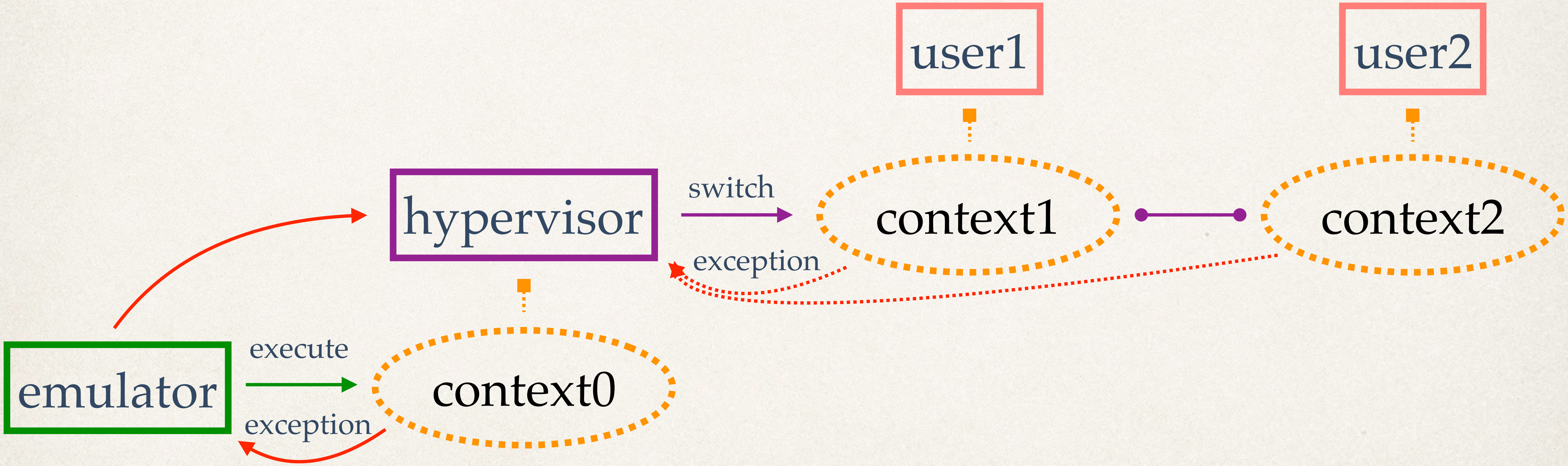


Synergy of Emulator & Hypervisor



```
while (1) {
  if (mix)
    from_context = mipster_switch(to_context, TIMESLICE);
  else
    from_context = hypster_switch(to_context, TIMESLICE);

  if (get_parent(from_context) != MY_CONTEXT) {
    // switch to parent which is in charge of handling exceptions
    to_context = get_parent(from_context);

    timeout = TIMEROFF;
  } else if (handle_exception(from_context) == EXIT)
    return get_exit_code(from_context);
  else {
    // TODO: scheduler should go here
    to_context = from_context;

    if (mix) {
      if (mslice != TIMESLICE) {
        mix = 0;

        timeout = TIMESLICE - mslice;
      }
    } else if (mslice > 0) {
      mix = 1;

      timeout = mslice;
    }
  }
}
```

Virtualization: Concurrency



Runtime Homework Ideas

- Processes and threads
- Locking and scheduling
- Atomic instructions and lock-free data structures
- Multicore support
- Large address spaces
- Conservative garbage collection

Thank you!