Project: OUROBOROS (snake eating own tail)

- python makes it easy to see byte code
  - use dis (for dissassembly)

-
```
def foo(n):
    return 42
```
  may think: if I see foo, just replace w/ 42

  But can't!

  it's a very dynamic language

- best effort approach: try to write compiler optimizations not to find every one, but to find some w/out destroying everything
- other dynamics: can change operators (can't change what + does, - does, etc)
  - C++ called operator overloading
- IR = low level representation (usually looks like control flow graph)
  - may want to do src → AST → IR, but actually want hybrid
  - widely used IR: static single assignment (SSA)
    - ↳ introduce new variables (x → x_prime)
      - ↳ essentially turning imperative lang into functional
      - ϕ nodes : too complicated to explain ☺
  - we will build a src-to-src compiler

    python code → do something → python code

- visitor pattern:

  ast_FunctionDef (   )

  - have a class, by default does noth for node
  - then have a class that's special, visitor visits it
  - we have to use to process ASTs (or could write function that walks the tree)

- 1st step: write simple visitors just to get feel for code (start small + get accustomed to this way of programming)
- straightforward sounding but lots of gotchas!
- +100 tests by Juan – so make/sha[illegible] lots of tests

ware a lot of

Last time talked about CISR v RISC, etc;

terms used in literature:
- superscalar architectures: multipe execution units
- super pipeling: multiple ins in eac execution; fetch, decod, execute

asynchronous

want pipeline as full as possible or get bubbles
emptying pipeline = flushing

FDIV Bug

- divide float in certain range w/ another float in another range (substantial error)
- error baked into chip!
- there are known errors in processors all the time, but not this bad
- led to recall of intel processor
- led to verification

32 bits / 32 bits → need 2^{64} tests

(1 bit / 1 bit → 4 tests)

64 / 64 → need 2^{128} tests

sidenote:
earth will be swallowed by sun ahhh

exhaustive testing doesn't work for large state spce

- formal verification (prove w/ math)
  - same dudes PhD dissertation
- model checking: inbtween formal ver + exh testing
  - automatic testing up to size n
    - collapses a bunch of "identical" states
    - standard approach

"testing can only reveal the presence of bugs but not their absence"
- justify formal verification
- but not exactly true

- What does TicTok do? what is formal spec? hard to specify things so formal verification is hard
  - what does it mean to be correct?
    - correct w/ respect to its specs
      - is the spec correct?
        - is the meta spec correct?

⋮

- qsort:
  - needs to terminate
  - if multiset input → output sequence

$\forall i, j \therefore i \le j : output[i] \le output(j]$

$i \ge 0$
$i < n$
$j \ge 0$
$j < n$

if tried to prov w/out all to left, it would fail

fairly simple spec

what if strings?, what does ≤ mean?, what is empty string? lower v. upper? spec becomes long
once spec is long, it's an "idutical prog" that's hard to read

Berger's Opinion:
why not high-level PL?.
- enrich w/:
  - post / pre conditions
  - invariants

No oracle!. No truth!
specs could be wrong

The Spec Problem
(spec not useful - not "clearly" correct)

The Oracle Problem
(where's the truth)

total correctness: prog = spec
partial correctness: prog properties are subset of spec
(ex: prog never % by 0)

---

explicitly parallel instruction computing (EPIC)

Intel + HP
org name: VLIW
very long instr word

ADD
ADD
MULT
SUB

VLIW:
[ADD ADD MULT SUB]

EPIC:
← if all ind, can run in parallel
only make big chunk if know can run in parallel
shift to the compiler the responsibility of identifying parallelism
↓ problem?
can't be done
(car 18 lane ex w/ ideal conditions)

plan:
new chip Itanium using EPIC
nickname: Itanic
unbelievable failure
- totally dif inst set (causes major compatibility problems)
- in C, this is impossible
- problem: aliasing (b/c of ptrs)
p and q can point to same object
combinatorial / state space explosion