Week 7.1

Thursday, May 9, 2024 10:21 PM

Stage 1: no globals, structs, function calls, or ptrs

stage 2: add globals

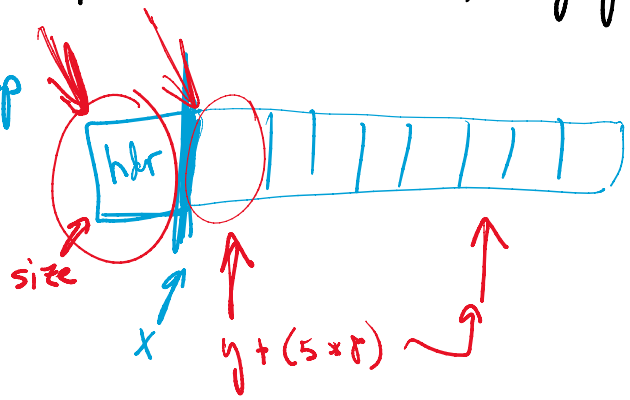
stage 3: add externs

stage 4: add functions

stage 5: add ptrs

lir insts: $load, $store, $alloc, $gep

x = $alloc op



- x = $load y
  - y's value is an address
  - get value at that address
  - store in x

- $store x op
  - x's value is an address
  - store value of op at that address
    - (not store to x)

- `x = $alloc op`
  - allocate op + 1 words
    - check op > 0 or panic
    - allocate using runtime library
  - store op into 1st word
  - store to x the address of the second word

  - compare op w/ 0, if not > then jump to `.invalid_array_length`
  - compute op + 1
  - call `_cflat_alloc`, passing op + 1
    - can clobber caller-save registers
    - call return value 'ptr'
  - store op to 'ptr'
  - store (ptr + WORDSIZE) to x

- `x = $gep y op`
  - compare op to 0, if < then jump `.out_of_bounds`
  - load value of (y - WORDSIZE); call it 'hdr'
  - compare op w/ hdr, if ≥ then jump .out_of_bounds
  - store (y + (op * WORDSIZE)) into x

Structs

additional lir : `$gfp`

fields are in alphabetical order

- `x = $gfp y fld`

  Let y : &st

  Let off be the offset fld in bytes

  add off to y and store in x

- `x = $alloc op`

  if x : &st then we're allocating (op * sizeof(st)) + 1

- `x = $gep y op`

  if y : &st then we store y + (op * WORDSIZE * sizeof(st)) into x

---

x86-64 | x64 | syntax: ATT ~~Intel~~

- 16 registers

  %r[...]

  ↳ 8-byte register

↑ 8-byte register

- %rsp stack pointer
  %rbp frame pointer
  %rax holds return values
  %rax, %rdx are used for division
  %rdi, %rsi, %rdx, %rcx, %r8, %r9
  are used for passing args.