isbool

isbool(true) ⇒ true
isbool(false) ⇒ true
isbool(4) ⇒ false

64-bits

R(true) 0x8000——01
R(false) 0x0000——01
R(3) 0x00——06
0x00——0[1000]

isbool(4)

```
mov rax, 0x0000000000000008
shl rax, 63
or  rax, 1
```

```
mov rax, 0x0000000000000008
and rax, 1
cmp rax, 1
je  ...
```

runtime errors

true + true ⇒ 1 ← reality
true + true ⇒ !! ← ideal

0x80——01 + 0x80——01 = 0x00——02

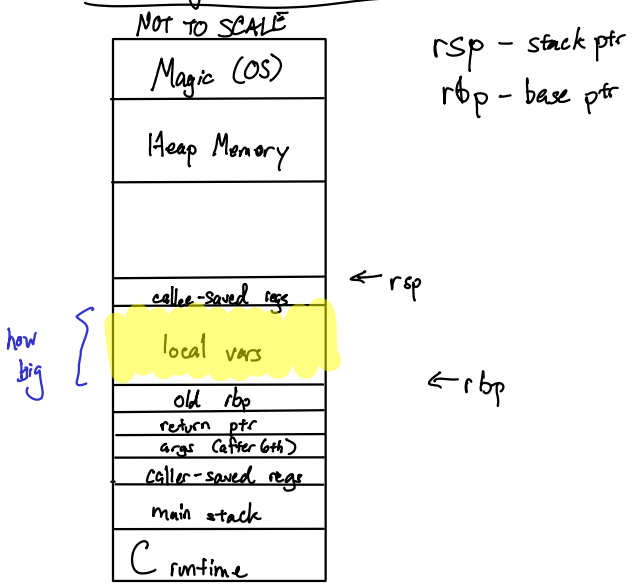
compiling $e_1 + e_2$

- compile e_1
- check
- store
- compile e_2
- check
- store
- load 1st
- op on 2nd

write helper functions!

Cardinal:
- resources/driver.c
- resources/error.c ← runtime errors
- resources/printer.c ← print(expr)

Calling Conventions in C



rsp - stack ptr
rbp - base ptr

Caller / Callee

POSIX 64-bit C Calling Conventions

1a. Push caller-saved registers: rax, rcx, rdx, r8, r9, r10, r11
1b. Store arguments
- First six args go into registers (rdi, rsi, rdx, rcx, r8, r9)
- Rest of args go onto stack in "reverse order"

1c. Call instruction
2a. Push rbp onto stack
2b. Copy rsp into rbp
2c. Subtract amt of memory = to my local & temp vars
2d. Save callee-saved registers
3. Do the thing
4a. Restore callee-saved regs
4b. Set rsp = rbp
4c. Pop old rbp into rbp
4d. ret
5a. Remove args from stack
5b. Restore caller-saved regs

*16-byte alignment

print(5) ⇒ 5

```
compile_expr
  match e with
  | EInt n -> mov rax, n*2
  | EAfter e' -> compile e' @ [add eax, 2]
  | ELet (x, e1, e2) ->
```

```
calc_expr_mem
  match e with
  | EInt n -> 0
  | EAfter e' -> calc_expr_mem e'
  | ELet (x, e1, e2) ->
      max (calc_expr_mem e1)
          (8 + calc_expr_mem e2)
```