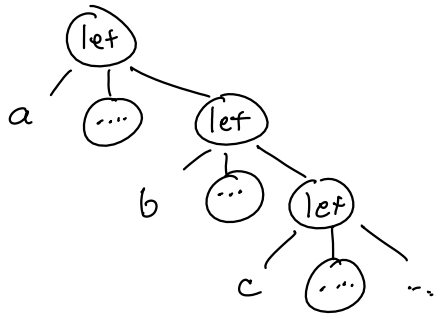
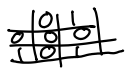
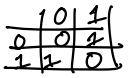
```
let a = ... in
let b = ... in
let c = .... in
...
```

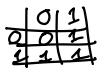


Bitwise Operations in x64

and	need both
or	need either
xor	need exactly one







flipping bits

tool for clearing bits: $b_3\ b_2\ b_1\ b_0$ AND $1\ 1\ 0\ 0$ = $b_3\ b_2\ 0\ 0$

setting bits: $b_3\ b_2\ b_1\ b_0$ OR $1\ 1\ 0\ 0$ = $1\ 1\ b_1\ b_0$

(¬) not

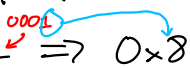
$b_3\ b_2\ b_1\ b_0$ XOR $1\ 1\ 0\ 0$ = $\neg b_3\ \neg b_2\ b_1\ b_0$

Lecture representation

true ≡ 0x800…01
false ≡ 0x000…01
int n ≡ 2·n

isbool(5) ⇒ false
isbool(true) ⇒ true
isbool(false) ⇒ true

0x0…0A ⇒ 0x0…01



0x8…01 ⇒ 0x8…01

0x0…01 ⇒ 0x8…01

$b_{63}\ b_{62} \cdots b_1\ b_0 \Rightarrow b_0\ 0 \cdots 01$

```
shl rax, 63
add rax, 1
```

$b_0\ 0 \cdots 00$

```
shl rax, 63
or rax, 1
```

Cardinal

	Semantics of Bluebird	Compilation of Bluebird	Semantics & Compilation Cardinal
true + true ⇒	error	1	runtime error

$0x80\cdots01 + 0x80\cdots01 \Rightarrow 0x02$ = Bluebird 1

Cardinal Plus:

1. compute left operand
2. store left operand
3. check left operand
4. compute right operand
5. store right operand
6. check right operand
7. actually add

exit with error code 1

an integer was expected (but we got something else)

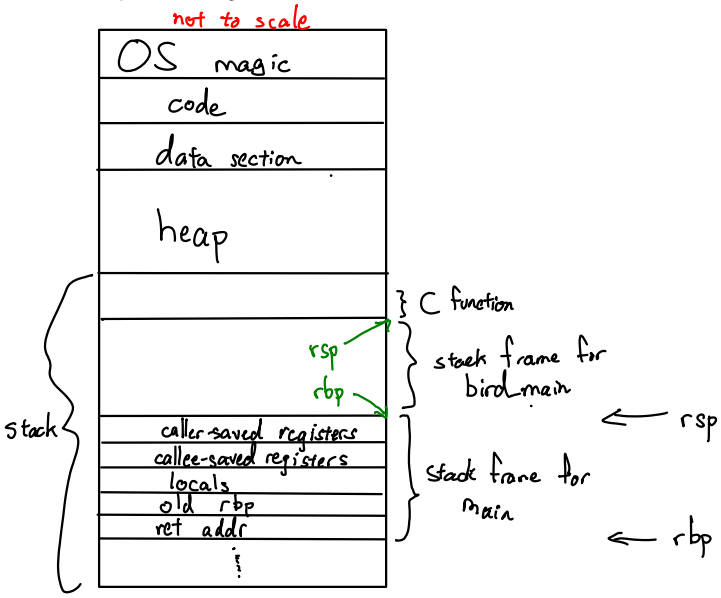
```
resources/
    driver.c  <- Auklet
    printer.c <- Bluebird
    error.c   <- Cardinal
```

void stopWithError(int)

How do I call a C function?

rsp – register: stack pointer

rbp – register: stack base pointer



POSIX 64-bit C calling conventions

Caller

1. a. push caller-saved registers (volatile registers)
   - rax, rcx, rdx, r8, r9, r10, r11
   b. position arguments
   - first six args go in: rdi, rsi, rdx, rcx, r8, r9
   - other args go on stack
   c. do call: `call lbl` 1. push ret addr onto stack 2. jumps label

Callee

2. a. push rbp
   b. mov rbp, rsp
   c. sub rsp, [...]
   d. push callee-saved regs: rbx, ~~rsp, rbp,~~ r12, r13, r14, r15
   e. do my thing
   f. a-d in reverse
   g. ret

Caller

3. a. pop args, saved regs from stack

<expr> ::= ... | print(<expr>)

print(5) → printing 5
evaluating to 5