Bluebird – binary representations / booleans

- every value : 64-bit
- 2^{64} combinations

language values	machine values (tag bit)
4 (…0100)	8 (…1000)
73	146
false (lecture mode)	1 (…0001)
true	129 (…10000001)

bad pick

00……00 – 0
00……01 – 2
……
01……00 – 1

---

isint(false) ⟹ false
isint(true) ⟹ false
isint(3) ⟹ true

Lecture mode: true 0…01000001
false 0…00000001

b_{63} b_{62} b_{61} …… b_2 b_1 b_0

	0	0	0	……	0	0	1
	b_{63}	b_{62}	b_{61}	……	b_2	b_1	b_0

selecting/clearing

and	0	1
0	0	0
1	0	1

setting

or	0	1
0	0	1
1	1	1

xor	0	1
0	0	1
1	1	0

flip

shr > binary shift
shl

Examples

```
mov rax, 5      ; 101
shr rax, 2      ; 00…001
shl rax, 3      ; 00…1000
```

isint(…)

	false	true	3
mov rax, …	00…01	00…10000001	00…110
and rax, 1	00…01	00…00000001	00…000
shl rax, 7	00…10000000	00…10000000	00…000
or rax, 1	00…10000001	00…10000001	00…001
xor rax, 128	00…00000001	00…00000001	00…10000001
	false	false	true

Cardinal

Bluebird

1 + 4 ⟹ 5

false + false ⟹ unspecified

Cardinal

1 + 4 ⟹ 5

false + false ⟹ exit code 1

1: expected an int

Addition:

1. Compute left operand
2. Check that it's an int
3. Store operand
4. Compute right operand
5. Check
6. Store operand
7. Load left
8. Perform op

Cardinal value

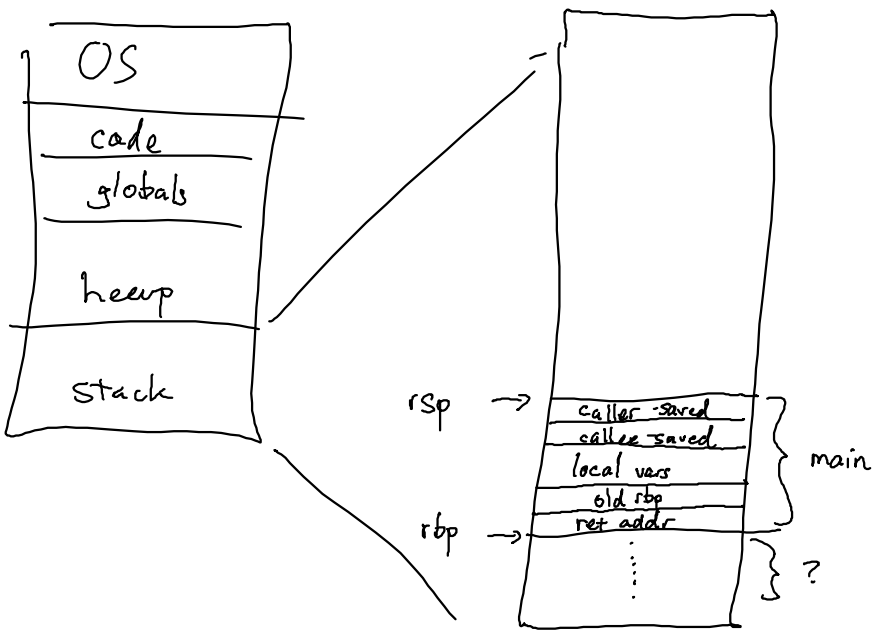
```
void printValue( )
void stopWithError( n )
```

exit code (machine)

Cardinal language

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

print(5) ⟹ 5 side-effect: printing 5



C calling conventions: x86-64

1a. Caller pushes caller-saved register

1b. Put arguments into registers/stack
- 1st arg always in RDI

1c. call instruction
- push rip
- jmp to code

2. Callee sets up stack frame
   a. push rbp
   b. set rsp and rbp to represent new stack frame
   c. callee pushes callee-saved
   d. does work

reverse of above