63-bit, 2's complement number

tag bit (1=num, 0=bool)

Value	**Representation (bits)**	**Representation** hex	decimal
		hex	**decimal**
9	0000 0000 0000 ... 0000 0000 000**1 0011**	0x000...00**13**	19
-2	1111 1111 1111 ... 1111 1111 1111 **1101**	0xFFF...FFF**D**	-3
33	0000 0000 0000 ... 0000 0000 __________	0x000...00__	___
true	**1**111 1111 1111 ... 1111 1111 1111 111**0**	0x**F**FF...FFF**E**	-2
false	**0**111 1111 1111 ... 1111 1111 1111 111**0**	0x**7**FF...FFF**E**	

true/false

62 bits for future value representations!

```
let rec e_to_is (e : expr) si env =
  match e with
    | ENum(n) ->




    | EBool(b) ->
```

```
#include <stdio.h>

extern int64_t our_code_starts_here()
                 asm("our_code_starts_here");

int main(int argc, char** argv) {
  int64_t result = our_code_starts_here();
  printf("%lld\n", result);
  return 0;
}
```

What will print from main for the program that's just the number 5?
For the program that's just the constant `true`?

What instructions should we produce for `(+ x y)` if we ignore errors? (Assume x at stackloc 1, y at stackloc 2)

```
| EPlus(e1, e2) ->
```

What instructions should we produce for `(+ x y)` if we want to *stop and print an error message* if x or y is a boolean?
What else might need to change?

```
let compile (program : string) : string =
  let ast = parse program in
  let instrs = e_to_is ast 1 [] in
  let all_is = (String.concat "\n" instrs) in
  (sprintf "
section .text
global our_code_starts_here
extern print_error_and_exit
our_code_starts_here:
  %s
  ret






" all_is)
```

```
#include <stdio.h>

extern int64_t our_code_starts_here() asm("our_code_starts_here");
```

What if we want to get input from *outside the program*?
First mechanism – command-line arguments. How to communicate to `our_code_starts_here` ?