Value	**Representation (bits)**	**Representation** hex	**Representation** decimal
	31-bit, 2's complement number / tag bit (1=num, 0=bool)		
9	0000 0000 0000 0000 0000 0000 000**1** **0011**	0x000000**13**	19
-2	1111 1111 1111 1111 1111 1111 1111 **1101**	0xFFFFFFF**D**	-3
33	0000 0000 0000 0000 0000 0000 ________	0x000000__	___
true	**1**111 1111 1111 1111 1111 1111 1111 111**0**	0x**F**FFFFFF**E**	-2
false	**0**111 1111 1111 1111 1111 1111 1111 111**0**	0x**7**FFFFFF**E**	214...
	true/false / 30 bits for future value representations!		

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


    | EBool(b) (* b is true or false *) ->
```

(Assume x at esp-4, which contains 0x0000013)

(Follow-up: What if esp-4 contained 0x7FFFFFFE?)

(add1 x)

In EAX at end?

(Assume x at esp-4, which contains 0x0000013)

(Assume y at esp-8, which contains 0xFFFFFFFD)

(> x y)

In EAX at end?

Example

```
cmp eax, ebx
jle done
```

```
and arg1, arg2
```
– bitwise and the args, store in arg1

```
or arg1, arg2
```
– bitwise or the args, store in arg1

If the contents of EAX are less than or equal to the contents of EBX, jump to the label *done*.

(Assume x at esp-4, which contains 0x0000013)

(Follow-up: What if esp-4 contained 0x7FFFFFFE?)

(add1 x)

In EAX at end?

(Assume x at esp-4, which contains 0x0000013)

(Assume y at esp-8, which contains 0xFFFFFFFD)

(> x y)

In EAX at end?

Example

```
cmp eax, ebx
jle done
```

If the contents of EAX are less than or equal to the contents of EBX, jump to the label *done*.

`and arg1, arg2` – bitwise and the args, store in arg1
`or arg1, arg2` – bitwise or the args, store in arg1

Value	**Representation (bits)**	**Representation** hex	**Representation** decimal
	31-bit, 2's complement number / tag bit (1=num, 0=bool)		
9	0000 0000 0000 0000 0000 0000 000**1** **0011**	0x000000**13**	19
-2	1111 1111 1111 1111 1111 1111 1111 **1101**	0xFFFFFFF**D**	-3
33	0000 0000 0000 0000 0000 0000 __________	0x000000__	___
true	**1**111 1111 1111 1111 1111 1111 1111 111**0**	0xFFFFFFF**E**	-2
false	**0**111 1111 1111 1111 1111 1111 1111 111**0**	0x**7**FFFFFF**E**	214...
	true/false / 30 bits for future value representations!		

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


    | EBool(b) (* b is true or false *) ->
```