Labs - collaborative
Projects - individual

- bitwise operators
- ~ , << , >>
  - masking]
  - get_bit(), set_bit()]
  - conv_uint32_to_binstr()]
  - sketch conv_binstr_to_uint32()
  - change parse_node_st and eval to use uint32_t
  - command line args ←
  - 2's complement]
  - ASR]

~	&	|	^	<<	>>	>>>
NOT	AND	OR	XOR	LSL	LSR	ASR

a	~a
0	1
1	0

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a|b
0	0	0
0	1	1
1	0	1
1	1	1

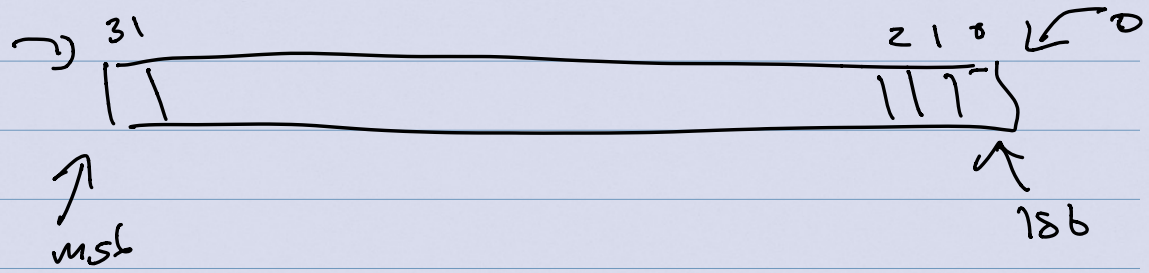
a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

<< LSL

0b0110 << 2 (2: num of bits)

= 0b1100 ←

uint32_t x;



0b0011 ⇒ 3

0b0011 << 1 ×2 $\times 2^1$

= 0b0110 ⇒ 6

0b0110 << 1 ×2

0b1100 ⇒ 12

0b0011 << 2 ⇒ $3 \times 2^2 = 3 \times 4$

0b1100 ⇒ 12

0b0011 << 3 $3 \times 2^3 = 3 \times 8$

$= 24$

0b11000

16 8

16+8=24

mul

lsl ✓

0b1100 0b1001

0b1100 & 0b1001 = 1000

0b1100 | 0b1001 = 1101

0b1100 ^ 0b1001 = 0101

>>

0b0110 >> 1

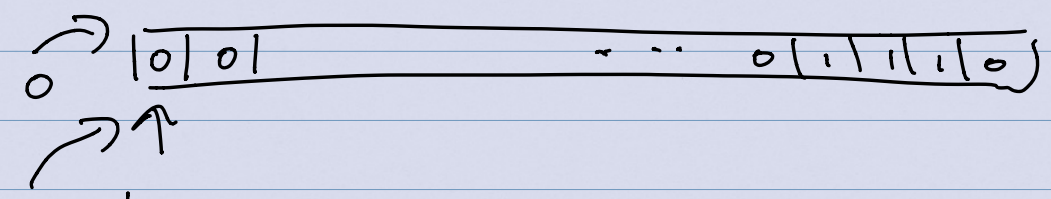
0b0011

uint32_t ux; int sx;

unsigned

ux = 0b1110

ux = ux >> 1



0b1100 >> 1 12

⇒ 0b0110 / by 2

↑ 6 6

---

masking 10

0b1010 1110 20

mask 0b1111 0000

& 1010 0000

↑ ⟶ >> 4

>> 4

0000 1010

0b1100 1010 1110

>> 4

0b0000 1100 1010

& 0b 0000 0000 1111

=> 0b 0000 0000 1010 10

uint32_t x = 1;

→ x = x + 2

→ x += 2

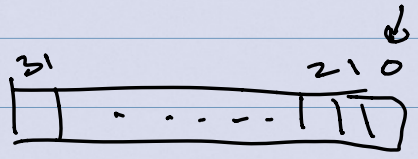
x = x & 2

x &= 2

x &= 0b10

get_bit()

uint32_t x;



$X_2 = 1$

$b = X_{11}$

uint32_t get_bit (int i, uint32_t v);

?

1 = get_bit (2, 061101)

0 = get_bit (1, 061101)

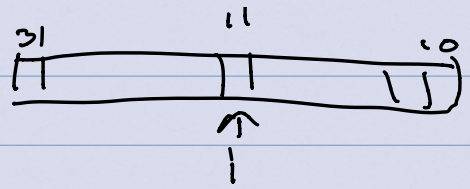
```
int get_bit (int i, uint32_t v) {
    int r;
    r = (v >> i) & 0b1
    return r;
}
```

i = 2

i = 1

i & v

```
  061101
& 060010
--------
  0000
```



~0b1100 1010

0b0011 0101

```
uint32_t set_bit (int i, uint32_t v) {
    uint32_t r;
    r = (0b1 << i) | v;
    return v;
}
```

OR 0b1001

0b0100 ←

1101

```
void set_bit (int, uint32_t *vp) {
    *vp = (0b1 << i) | *vp;
}
```

---

input "0b1010" => uint32_t Sketch

output uint32_t => str *

13 = "0b1101"

0b1101

"0b0000 ... 0000 1101"

conv.c → conversion functions

```
void conv_uint32_to_binstr(uint32_t v, 32bit 4bytes
                           char* str)
{
    int i,j,bit;

    str[0] = '0';
    str[1] = 'b';
    j = 2;

    for(i=31; i>=0; i--) {
        bit = get_bit(i,v);
        bit = (v>>i) & 0b1 =

        str[j] = '0' + bit;
        j += 1;
    }
    str[j] = '\0';
}
```

i: 31 … 0

str[0] str[1] str[2]

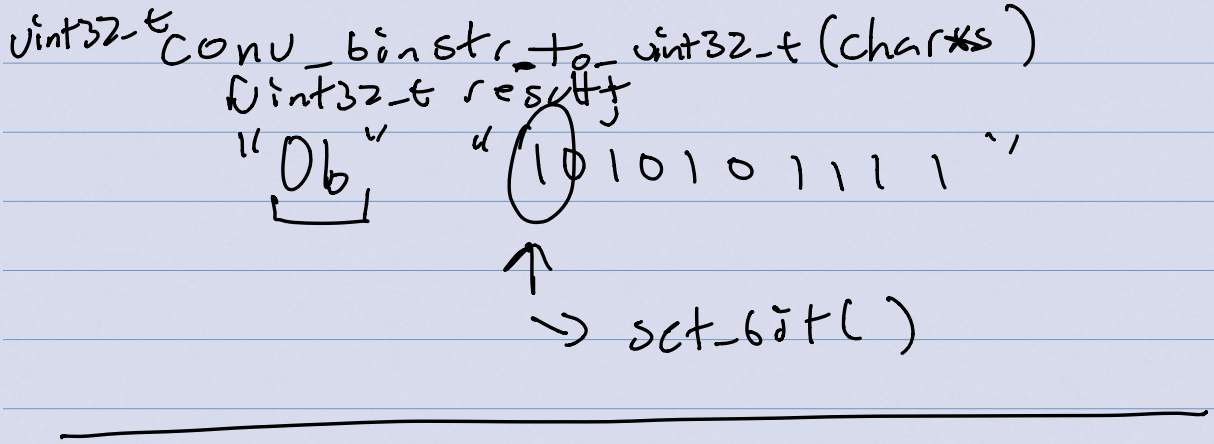
str = "0b10010"

"hello"

"0b" j

"0b0" j

"0b00" j ----->


uint32_t conv_binstr_to_uint32_t (char*s)
{uint32_t result
"0b"
"10101011 1"
-> set_bit( )