Labs - collaborative

Bitwise operators Projects - individual

masking

get_bit(), set_bit()

conv_uint32_to_binstr()

change parse_node_st and eval to use uint32_t

command line args

2's complement] Tue

ASR

~ & | ^ << >> >>-

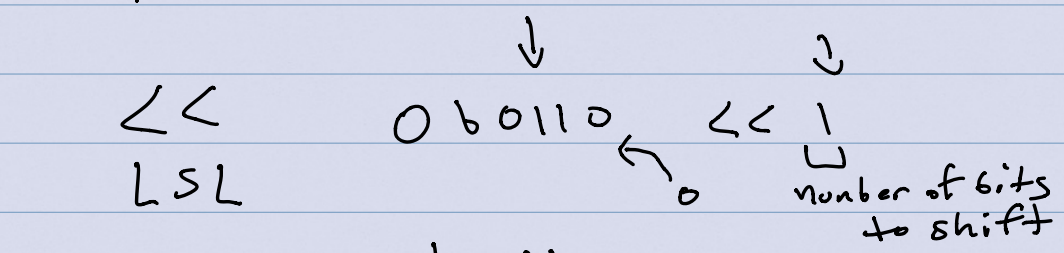
NOT AND OR XOR LSL LSR ASR

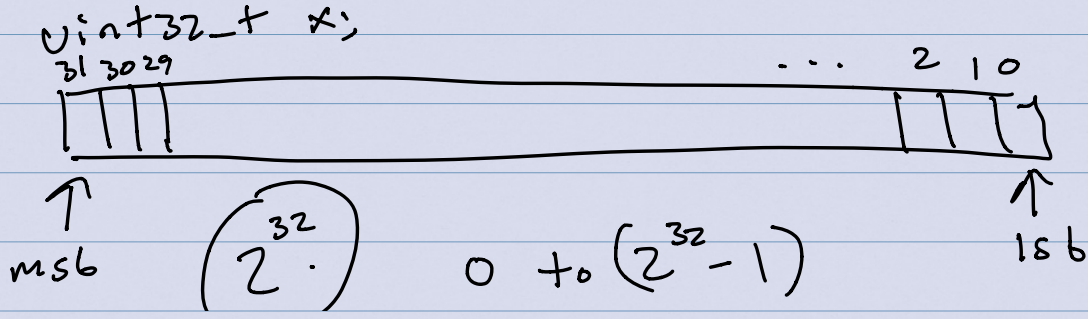
a	~b
0	1
1	0

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





0b0011 → 3

×2

0b0011 << 1

= 0b0110 → 6

×2

0b0110 << 1

= 0b1100 → 12

0b0011 << 2

0b1100 → 12

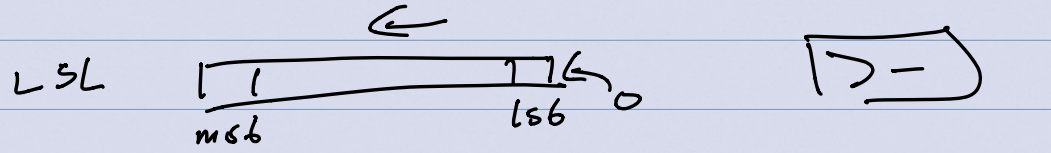
bitwise ops

0b1100 0b1001

0b1100 & 0b1001 = 1000

0b1100 | 0b1001 (OR) = 1101

0b1100 ^ 0b1001 (XOR) = 0101



x >= 1

Masking

0b 1010 1110

0b 1010 1110

& 0b 1111 0000 mask

1010 0000 >> 4

= 0000 1010 => 10

0b 1100 1010 1110

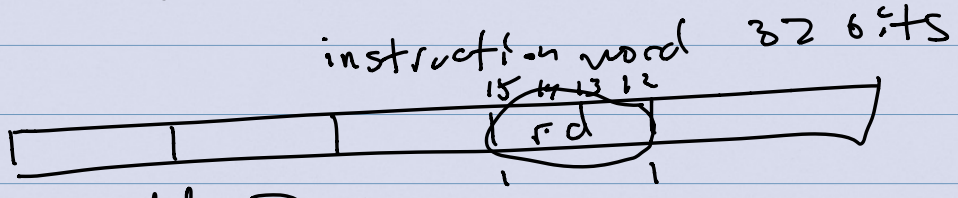
0b 0000 1111 0000

0b 1100 1010 1110 >> 4

0b 0000 1100 1010

& 0b 0000 0000 1111

0b 0000 0000 1010



add r0, r1, r2

~

~(0b10101100)

= 0b0101 0011

↑ Unsigned int

uint32_t x = 1;

x = x + 1;

x += 1;

x = x & 2

x &= 2;

x & = 0b10

Scanner

"0b1101"

↓

tk.value = "1101"

parse_operand()

TK_BINLIT

convert → bin str → uint32_t

of

int val

```
int get_bit(int i, uint32_t v);

1 = get_bit(2, 0b1101)
0 = get_bit(1, 0b1101)
```

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

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

= set_bit_on(3, 0b11000001)

= 0b11001001

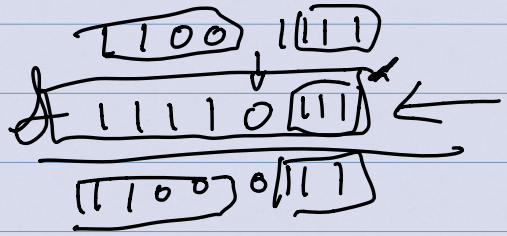
0b0000 1000
0b0000 0001

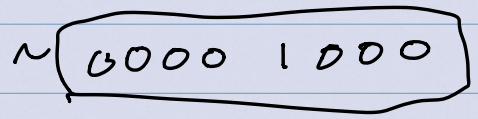
0b11000001
0b0000 1000
0b11001001

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

set_bit_off (3, 0b1100 1111)

= 0b1100 0111





~(a & b)

~a | ~b

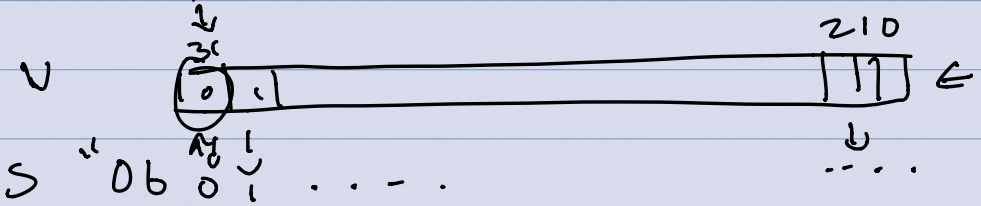
uint32_t set_bit (int i, uint32_t v, int b);

0 1

---

lab4 input "0b1010" ⇒ uint32_t

output uint32_t ⇒ "0b1010"



```
void conv_uint32_to_binstr(uint32_t v,
                           char *s)
{
    int i, j, bit;

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

    for (i=31; i>=0; --i) {
        bit = get_bit(i, v)
        bit = (v >> i) & 0b1
        if (bit == 0) {
            s[j] = '0'
        } else {
            s[j] = '1'
        }
        j += 1;
    }
    s[j] = '\0';
}

uint32_t conv_binstr_to_uint32(char *s)
{
    uint32_t r = 0;
    "0b" "1010 1100" ...
}
```

setb:f()