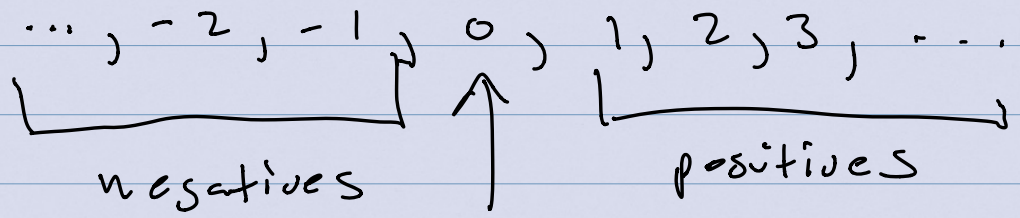
Negative Numbers

int whole numbers



3.781 $\quad -1.5 \quad 3.1 \times 10^{-2}$

Integers

n-bit numbers

3-bit numbers (unsigned)

Bin	Dec	
000	0 ←	n-bits
001	1	
010	2	2^n different values
011	3	
100	4	0 to $2^n - 1$
101	5	0 to $2^3 - 1$
110	6	0 to 8 - 1
111	7 ←	0 to 7

Negatives 1/2 positive 1/2 negative

Signed Magnitude

Bin	Dec	
000	0	*
001	1	
010	2	
011	3	
100	-0	*
101	-1	
110	-2	
111	-3	

Pluses:
1) simple
2) easy to determine if value is neg

if (x == 0) { x == +0 || x == -0

3

-1 + 3 = 2 101 ( )

1 1 1
101 (-1)
+ 011 (3)
000 (0) X

1+1 = 10

001 ⇒ 110

+1 -1

Two's Complement

Bin	Dec	
000	0	← only 1 zero
001	1	
010	2	
011	3	
→ 100	-4	←
101	-3	✗
110	-2	✗
111	-1	

↑

n-bit

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

convert pos to neg

neg to pos

pos $(010)_{(2)}$ What is the 2's comp neg representation

Invert Add One

2 [010] ~010 = 101 + 1 = [110] -2

3 [011] ~011 = 100 + 1 = [101] -3

tilde ~

neg → pos

(−2)
110   ~110 = 001 + 1 = 010
2

(−3)
101   ~101 = 010 + 1 = 011
3

2's comp arithmetic

−1 + 3

1+1+1 = 11

```
 1 1 1
   111  (−1)
+  011  (3)
  ----
   010  (2)
```

2 − 4 = 2 + (−4)   100 ~100 = 011 + 1 = 100

```
   010  (2)
+  100  (−4)
  ----
   110  (−2)?
```

001 + 1 = 010
2

2's comp

$-(2^{n-1})$ to $(2^{n-1} - 1)$

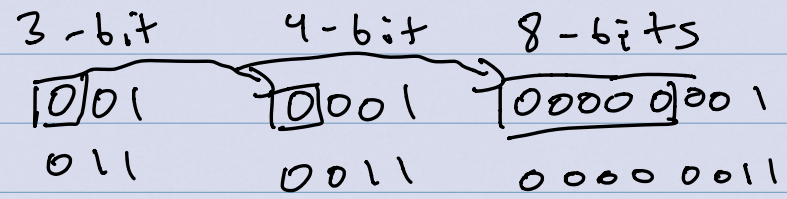
4 bits

-2^3

-8 to 7

Two's Comp: Sign Extension and ASR

pos



neg?

3-bit (-2) 4bit 0010 ⇒ 1101 + 1 = 1110

110 → 1110

8 bit?

0000 0010 ⇒ 1111 1101 + 1

1111 1110

	unsigned	signed
111	7	-1

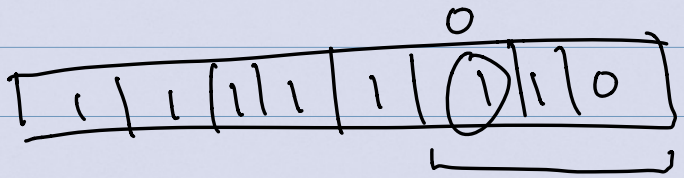
int -2

(x < 4) == true int

unsigned int

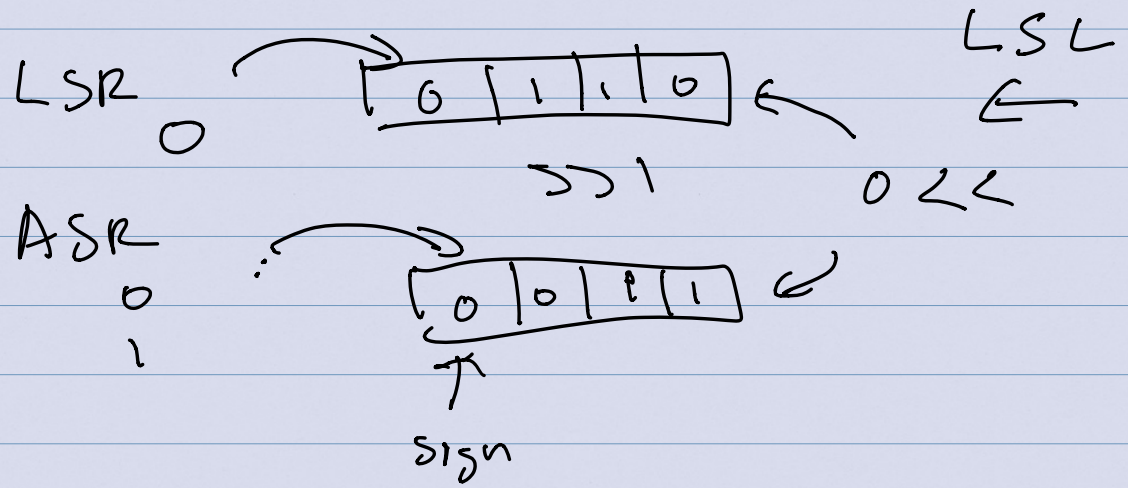
int unsigned int

Sign extension



ASR Arithmetic Shift Right

LSR Logical Shift Right



uint32_t v1;

v1 >- OP_ASR

v1 >- v2

result = ((int) v1) >> v2

```
int x = -4;
int y = -8;

unsigned int ux;
unsigned int uy;

ux = (unsigned int) x;
uy = (unsigned int) y;

int r
        ↓
r = (int) ( ux + uy )

r = -12
```

"0110"
↑
ch

ASCII

49	48	1
48	48	0

```
v = ch - '0'
```

"123"

```
v = ch - '0'
```

A a
↑ ↗

"AB12" "1/"

ch = tolower(ch)

// if (ch >= '0' && ch <= '9')

v = ch - '0'

else if (ch >= 'a' && ch <= 'f')

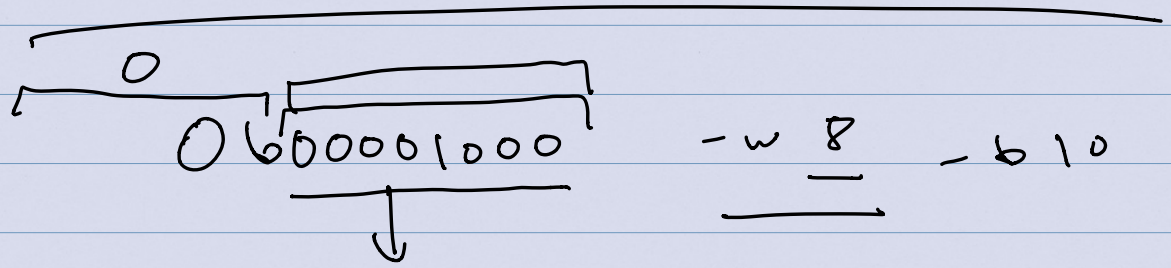
v = (ch - 'a') + 10

'a'

0

0xab3

0xAB3



32bit uint32_t

res = res & 0xFF

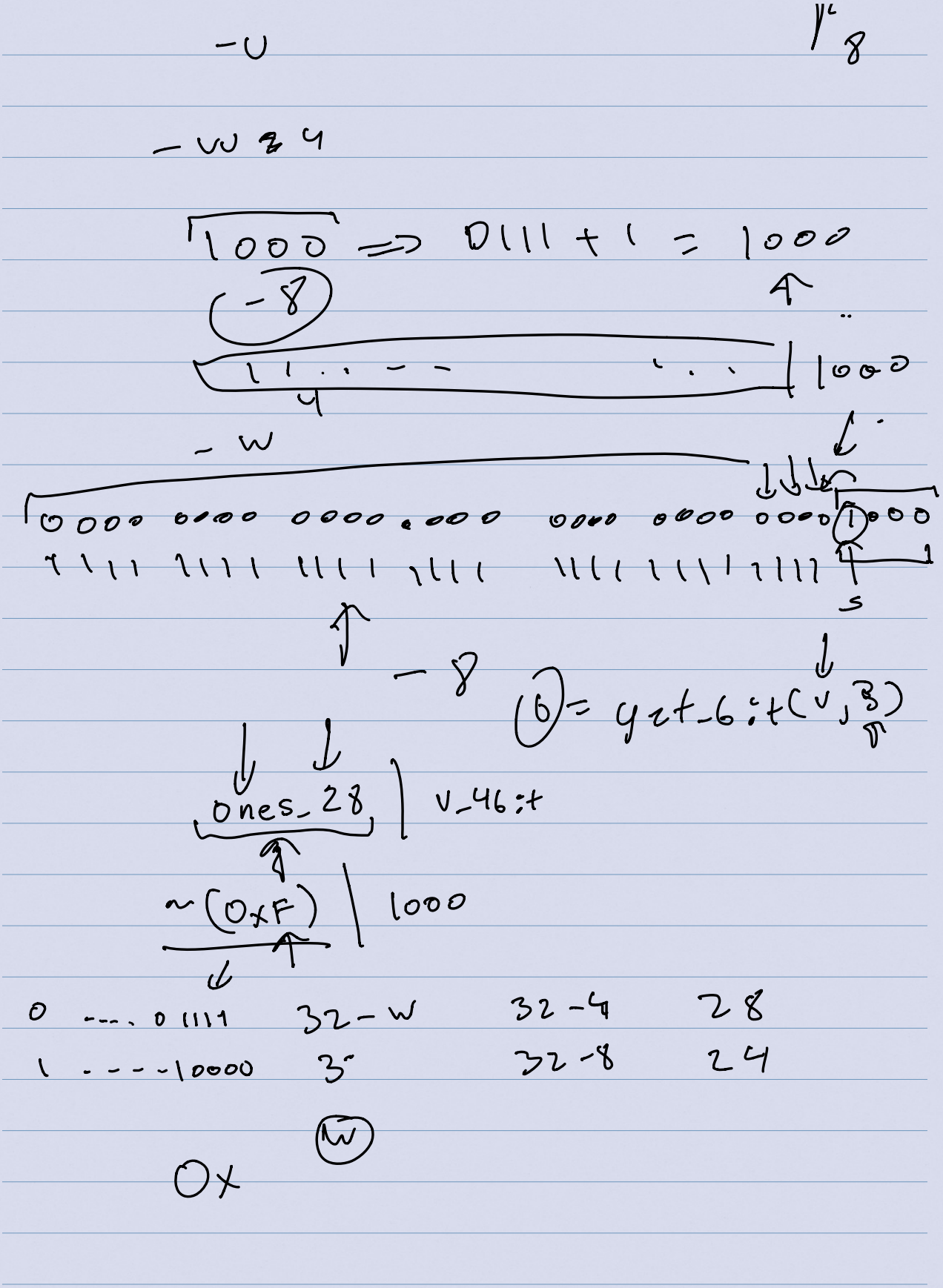
res =

32 bit 8 =

0000 0000 0000 0000 0000 0000 0000 1000

8 bit 8

0000 1000


-U
-W & 4
1000 ⇒ 0111 + 1 = 1000
-8
1000
-W
-8
(6) = y2t_6:t(v, 3)
ones_28
v_46:t
~(0xF)
1000
0 ... 0 1111 32-W 32-4 28
1 ... 10000 3 32-8 24
W
0x