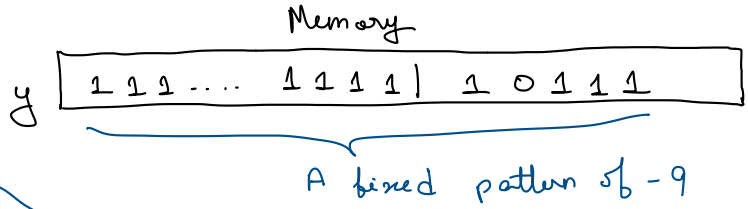
2a - Signed and Unsigned No. in Memory

Monday, 27 July 2026 03:37

int is 32 bits

```
int y = -9;
```




```
printf ("%d ", y);
printf ("%u", y);
```

What is the output?

$(9)_{10} = (1001)_2$

but since int is 32 bit, we need 32 bits in binary to represent $(9)_{10}$.

$= (000.....001001)_2$

So -9 is 2's comp of 9

$= (111.... 110110 + 1)_2$

$= (111.... 110111)_2$

① `printf ("%d", y);`

`%d` prints the value in memory as a signed number

o/p : -9

② `printf ("%u", y)`

`%u` prints the no. as unsigned

So it converts $(111\ldots110111)_2$ in decimal.

NOTE: Printf does not make use of the type of variable.

Extension

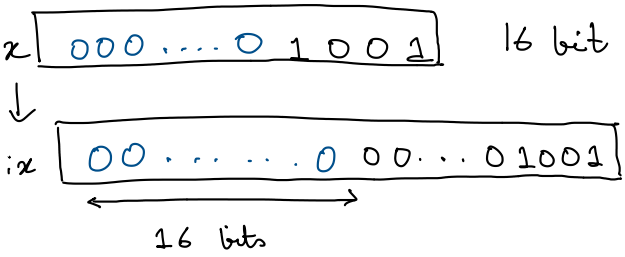
Copying lower bit to higher.

1. short int x = 9;

   → 16 bit

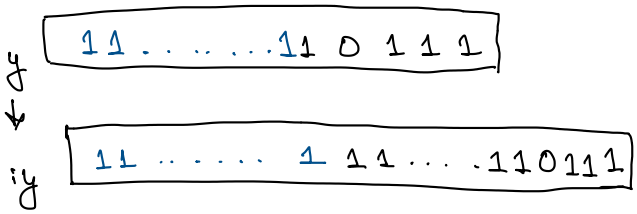
   int ix = x;

   → 32 bit



2. short int y = -9;

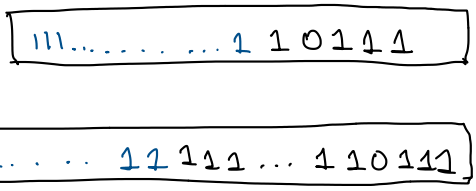
   int iy = y;



3. Extending Signed to unsigned

short int y = -9;

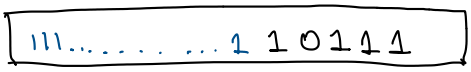
unsigned int iy = y;



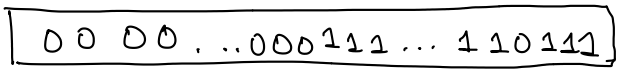
Since source is signed, the destination being unsigned does not matter.
So 1's are placed.

Q. Extending Unsigned to Unsigned.

unsigned short int y = -9;



unsigned int iy = y;

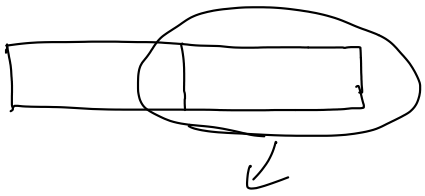


Extension always depends on RHS (Source)

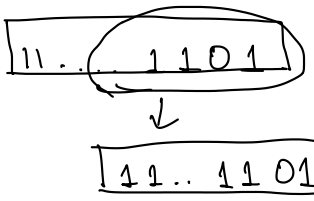
Promotion always according to the source and not the destination.

Truncation:

- Regardless of source or destination signed / unsigned, truncation just truncates
- This can cause no. to change value or sign.



eg: int x = -9;



short y = x;