Example: Add 6-bit numbers -17_{10} and 23_{10}

From before: $-17_{10} = 101111_2$
$23_{10} = 010111_2$

```
   1 1 1 1 1
   1 0 1 1 1 1
   0 1 0 1 1 1
(1) 0 0 0 1 1 0
```

6-bit solution

Important: In unsigned addition, this carry-out means an overflow has occured

2's-complement addition: always ignore end carry! (may or may not indicate overflow)

2's-complement range for n-bit numbers

n	range
4	$-8 \le N \le 7$
16	$-32768 \le N \le 32767$
32	$-2,147,483,648 \le N \le 2,147,483,647$

Overflow: Only possible if adding numbers of the same sign

Example: $23_{10} + 23_{10}$ using 6 bits, Express answer in signed decimal notation (e.g., -10, +27, etc..)

One way to recognize overflow:

```
 0  1 1 1 1
    0 1 0 1 1 1
    0 1 0 1 1 1
(0) 1 0 1 1 1 0
```

Answer: 101110

whoops! looks negative

Answer is clearly wrong. Convert to signed decimal anyway.

- negative so perform 2's-complement negation to find its' magnitude

$$101110 \longrightarrow \begin{array}{r} 010001 \\ +\quad 1 \\ \hline 010010 \end{array} = 16 + 2 = 18_{10}$$

(answer = -18_{10})

So $23_{10} + 23_{10} = -18$? (wrong!)

Overflow <u>never occurs</u> when adding numbers of <u>opposite</u> sign

<u>Another way to recognize overflow</u>

$$\begin{array}{cr} 0 & 1\,1\,1\,1\,1\,1 \\ & 010111 \\ & +\,010111 \\ (0) & 101110 \end{array} \qquad \begin{array}{l} (+23) \\ (+23) \\ (-18) \end{array}$$

Check last two carry bits

If the carry-in and carry-out on the <u>last bit</u>

- are opposite $\longrightarrow$ overflow!
- are the same $\longrightarrow$ no overflow

<u>Some important codes</u>

So far, we have discussed <u>binary codes</u>, namely unsigned binary, sign/magnitude, and 2's-complement.

Other important codes include ASCII for text, BCD codes for base-10 numbers, Gray codes.

<u>Binary-Coded Decimal (BCD) codes</u>

These codes are useful in human-machine interfaces — a convenient way of representing <u>decimal</u> numbers

Decimal number	BCD code
0	0000
1	0001
2	0010
3	0011
⋮	⋮
8	1000
9	1001

← each BCD code word is 4 bits

codes 1010 – 1111 not used; invalid BCD codes

Example: Represent the decimal number 2468_{10} in BCD

We have 4 digits, and each one gets a 4-bit code

$$\underbrace{0010}_{2}\ \underbrace{0100}_{4}\ \underbrace{0110}_{6}\ \underbrace{1000}_{8}$$

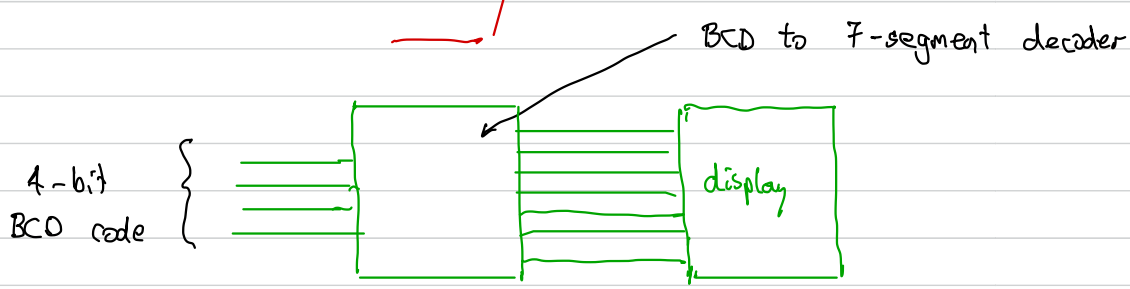
A very common device for displaying decimal numbers is called a <u>seven-segment display</u>

← a segment; typically a LED that can be switched on and off

Found in some scoreboard displays, clock radios, appliances.
- forms digits by appropriately illuminating segments.

e.g.,

(five)



Gray code

A very important code. A 3-bit Gray code looks like

Number	Binary code	Gray code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100