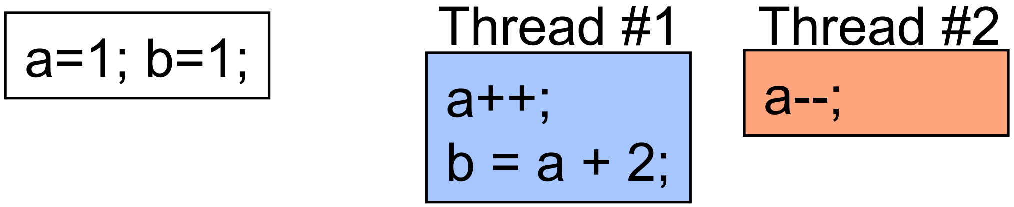


```
a=1; b=1;
```

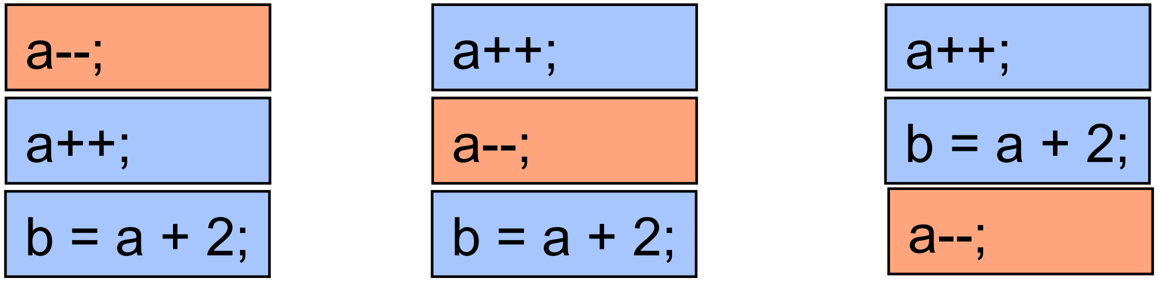
Thread #1

```
a++;
b = a + 2;
```

Thread #2

```
a--;
```

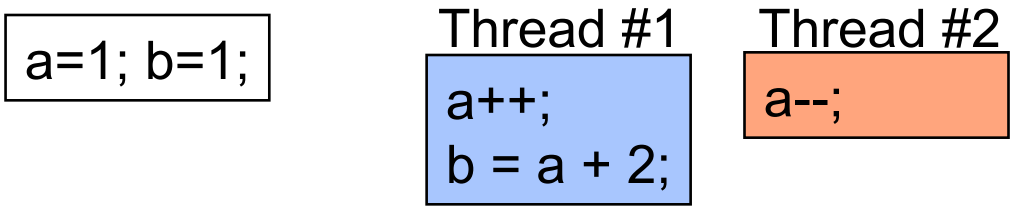
- First thing to do: come up with all possible interleaving of the instructions assuming that all instruction is executes entirely without being interrupted



```
a--;
a++;
b = a + 2;
```

```
a++;
a--;
b = a + 2;
```

```
a++;
b = a + 2;
a--;
```



```
a=1; b=1;
```

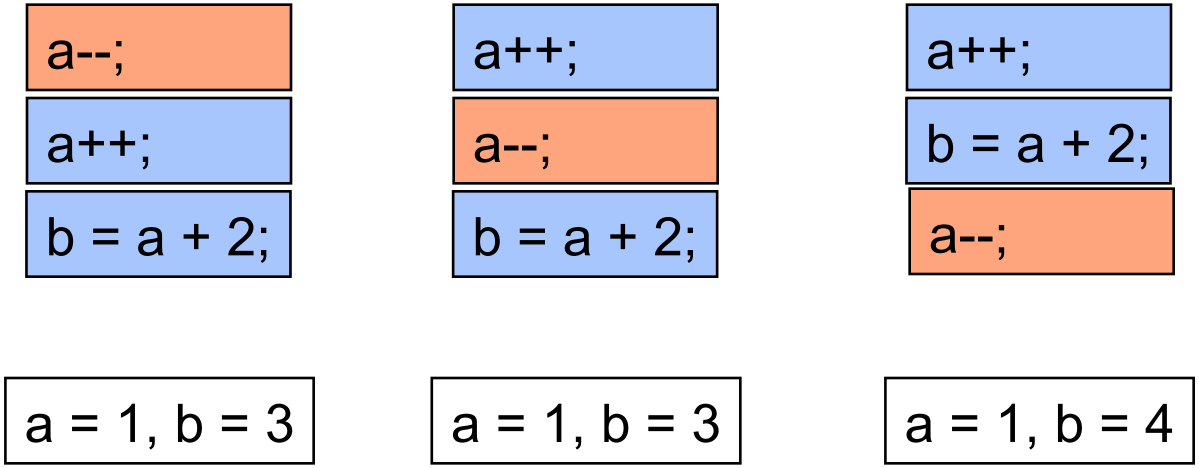
Thread #1

```
a++;
b = a + 2;
```

Thread #2

```
a--;
```

- First thing to do: come up with all possible interleaving of the instructions assuming that all instruction is executes entirely without being interrupted



```
a--;
a++;
b = a + 2;
```

```
a++;
a--;
b = a + 2;
```

```
a++;
b = a + 2;
a--;
```

a = 1, b = 3

a = 1, b = 3

a = 1, b = 4

```
a=1; b=1;
```

Thread #1

```
a++;
b = a + 2;
```

Thread #2

```
a--;
```

- Second thing to do: lost updates
  - Each line of code consists of multiple “hardware” instructions
- In this case: bad interaction between “a++” and “a--”
  - Result: a = 2
    - “a--” reads value 1, computes 0, gets interrupted
    - “a++” reads value 1, computes 2, gets interrupted
    - “a--” writes value 0
    - “a++” writes value 2, overwriting the 0
  - Result: a = 0
    - Same as “a=2” just different order
  - Result: a =1
    - Everything went well, without lost update
- We end up with two new possible output:

a = 0, b = 2

a = 2, b = 4

```
a=1; b=1;
```

Thread #1

```
a++;
b = a + 2;
```

Thread #2

```
a--;
```

a = 1, b = 3

a = 1, b = 3

a = 1, b = 4

- Output produced for all possible interleaving of lines of code
  - Can be considered a bug or not depending on what you application does
  - An application must not necessarily be 100% deterministic to be ~~correct~~ acceptable
    - Input could be random anyway

a = 0, b = 2

a = 2, b = 4

- Output produced due to the lost update problem
  - Typically considered a bug because a has a value different from 1 after “a++” and “a--” in the code, and b can take value 2 which likely makes no sense