Today

- Learning outcomes
  - Describe what a processor does when we makes an incorrect prediction
- How we'll get there
  - Review how/when a mis-predict happens.
  - Practice rearranging loops to improve prediction behavior.

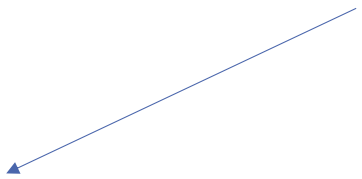
Recall

- To mitigate control hazards due to conditional branches, we said that we could make a guess (e.g., prediction) and then keep executing based on that prediction.
- But: What do we do if that prediction is wrong?

Mis-Prediction

- What happens if you are wrong?
- Consider:

```
        irmovq 0x1, %rax
        irmovq 0x1, %rbx
        subq %rax, %rbx
        je skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
        <more instructions here>
skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



Let's assume that our prediction algorithm is "never taken"

You have two bad instructions in the pipeline!

- That is, instructions you should not have executed!
- So what do you do???

You cancel or squash or quash them!

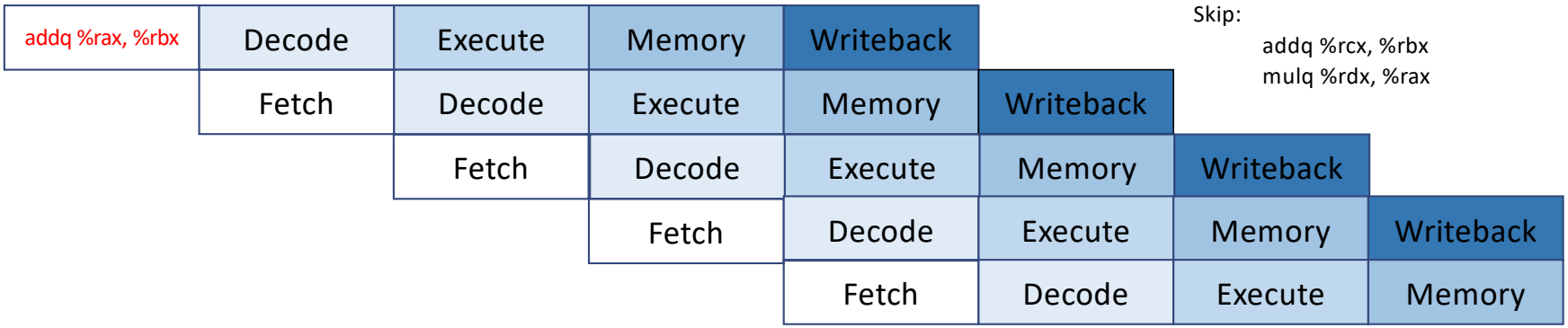
So:

- In the best case, conditional branches incur no penalty
- In the worst case (where you mispredict), it's no worse than if you'd done no prediction)

Branch Mis-Predicts (1)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -10

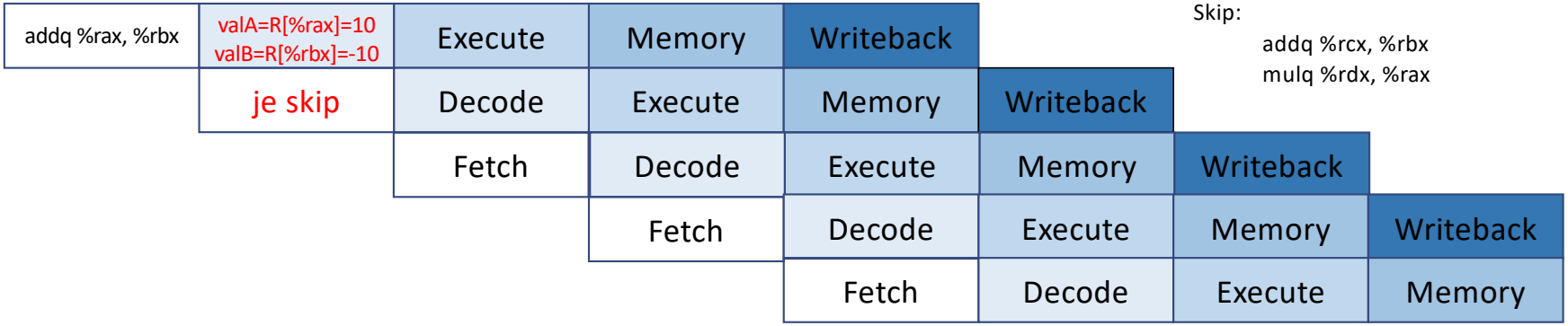
```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



Branch Mis-Predicts (2)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -10

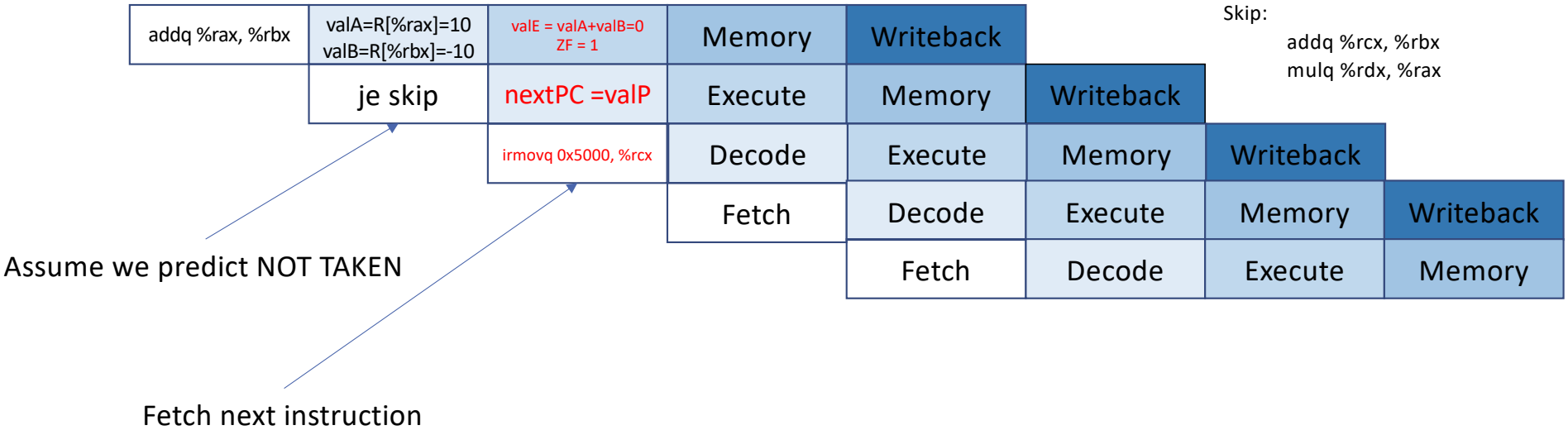
```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



Branch Mis-Predicts (3)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -10

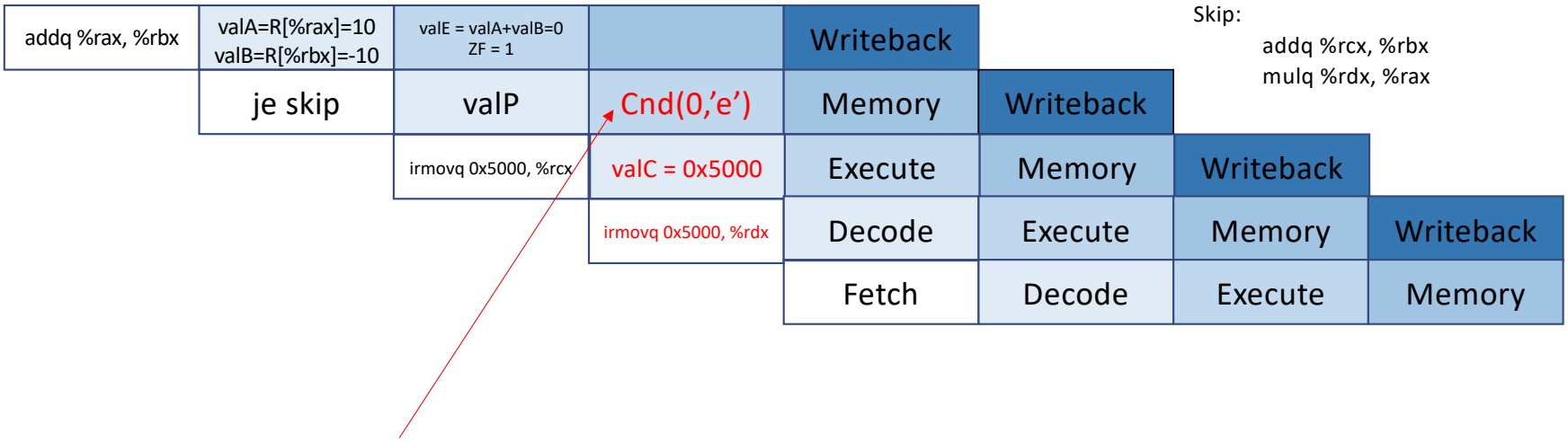
```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



Branch Mis-Predicts (4)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -10

```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```

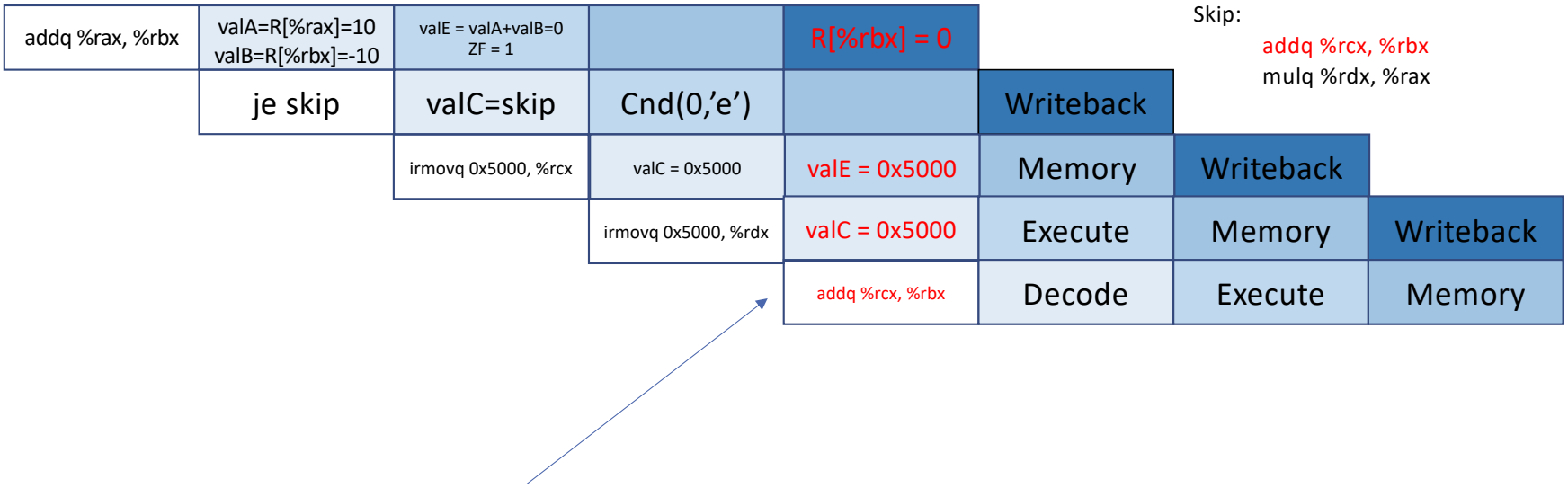


Uh oh! We can now evaluate the condition, and it turns out **we should have jumped**!

Branch Mis-Predicts (5)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -10

```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```

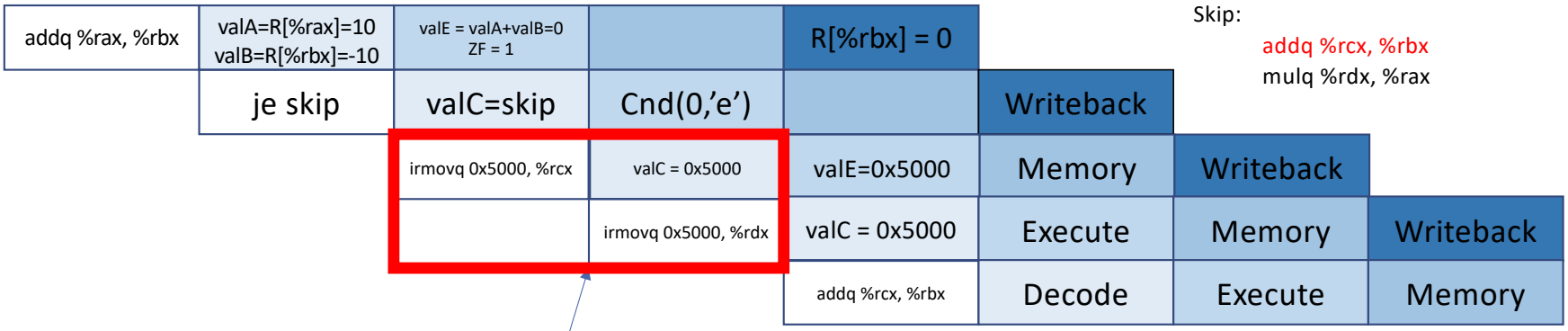


OK – so now we are executing at the place we should have been executing after the jump

Branch Mis-Predicts (6)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = 0

```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



But what about these instructions???

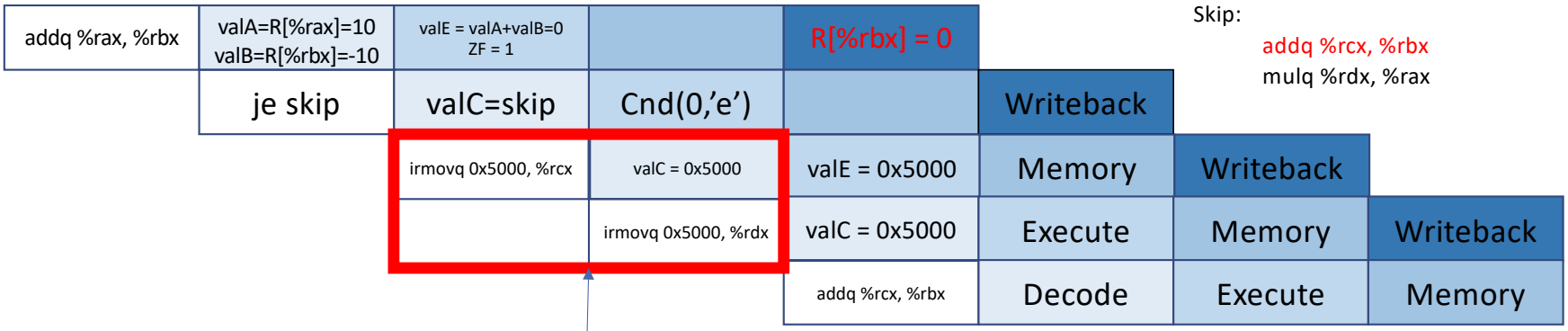
Bad news: You have started executing two instructions that you should not have.

Good news: They haven't modified any state that anyone has "seen."

Branch Mis-Predicts (7)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -9

```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



We will cancel/squash/quash them!
Make sure that no visible changes get made:

- No writeback
- No writes to memory
- No update of condition codes

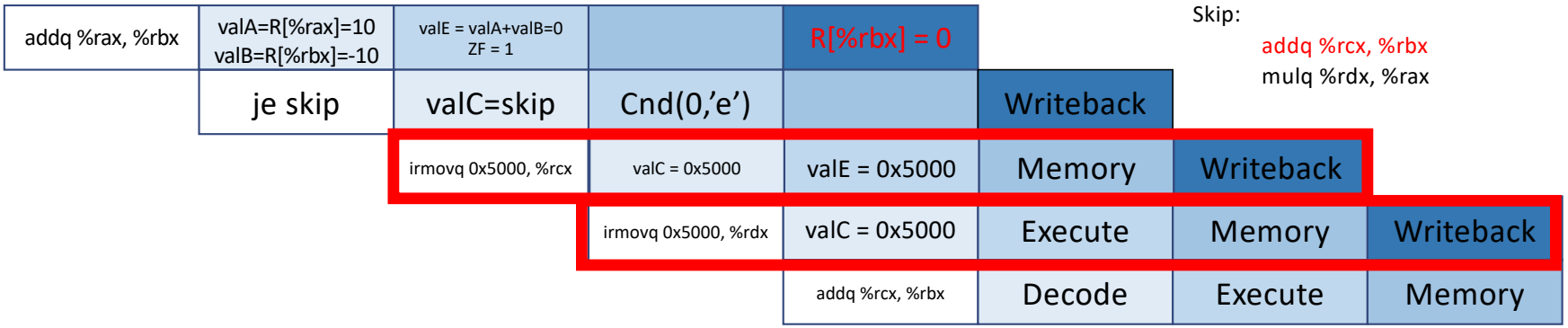
Bad news: You have started executing two instructions that you should not have.

Good news: They haven't modified any state that anyone has "seen."

Branch Mis-Predicts (8)

R[%rcx] = 1, R[%rdx] = 1
R[%rax] = 10, R[%rbx] = -9

```
        addq %rax, %rbx
        je Skip
        irmovq 0x5000, %rcx
        irmovq 0x5000, %rdx
Skip:
        addq %rcx, %rbx
        mulq %rdx, %rax
```



+1

9 cycles/3 instructions completed = 3 CPI