CS 554 – Advanced Topics in Concurrent Computing Systems

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Welcome to the Course!

- This is CS 554, Advanced Topics in Concurrent Computing Systems
- I'm glad you're here!
- Meets MW 11:00 AM – 12:15 PM in CDLRC 2407
- My office hours: Tuesday 10 AM, Thursday 2 PM, and by appointment, in CDLRC 4450 and on Zoom via Blackboard
  - Office hours are great for homework help, or just to say hi!

Course Information

- Professor: William Mansky (he/him) (mansky1@uic.edu)
- Website: https://www.cs.uic.edu/~mansky/teaching/cs472/sp25/
- Anonymous in-class questions: https://pollev.com/wmansky771
- In-person lectures, in-class exercises
- Discussion board on Piazza, assignments via Gradescope (entry code J6GYJV)

Asking questions

- In class: raise your hand anytime
- You can ask questions anonymously with PollEverywhere (https://pollev.com/wmansky771)
- On Piazza
  - Can ask/answer anonymously
  - Can post privately to instructors
  - Can answer other students' questions
- In office hours

- If you have a question, someone else probably has the same question!

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Concurrent programming

```
int add1(void* arg){
      int* p = (int*) arg;
      int val = *p;
      *p = val + 1;
}

int main(){
      int* x = malloc(sizeof(int)); *x = 0;
      int* y = malloc(sizeof(int)); *y = 0;
      thrd_t *t;
      thrd_create(t, add1, x);
      add1(y);
}
```

Concurrent programming

```
int val = *x;
*x = val + 1;
```

```
int val = *y;
*y = val + 1;
```

x	y
0	0

Memory

Concurrent programming

```
    int val = *x;
→   *x = 0 + 1;
```

```
→   int val = *y;
    *y = val + 1;
```

x	y
0	0

Memory

Concurrent programming

```
    int val = *x;
→   *x = 0 + 1;
```

```
    int val = *y;
→   *y = 0 + 1;
```

x	y
0	0

Memory

Concurrent programming

```
int val = *x;
*x = 0 + 1;
```

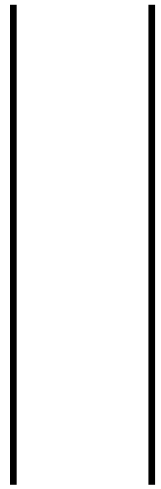
```
int val = *y;
*y = 0 + 1;
```

x	y
1	0

Memory

Concurrent programming

```
int val = *x;
*x = 0 + 1;
```

```
int val = *y;
*y = 0 + 1;
```


x	y
1	1

Memory

Questions

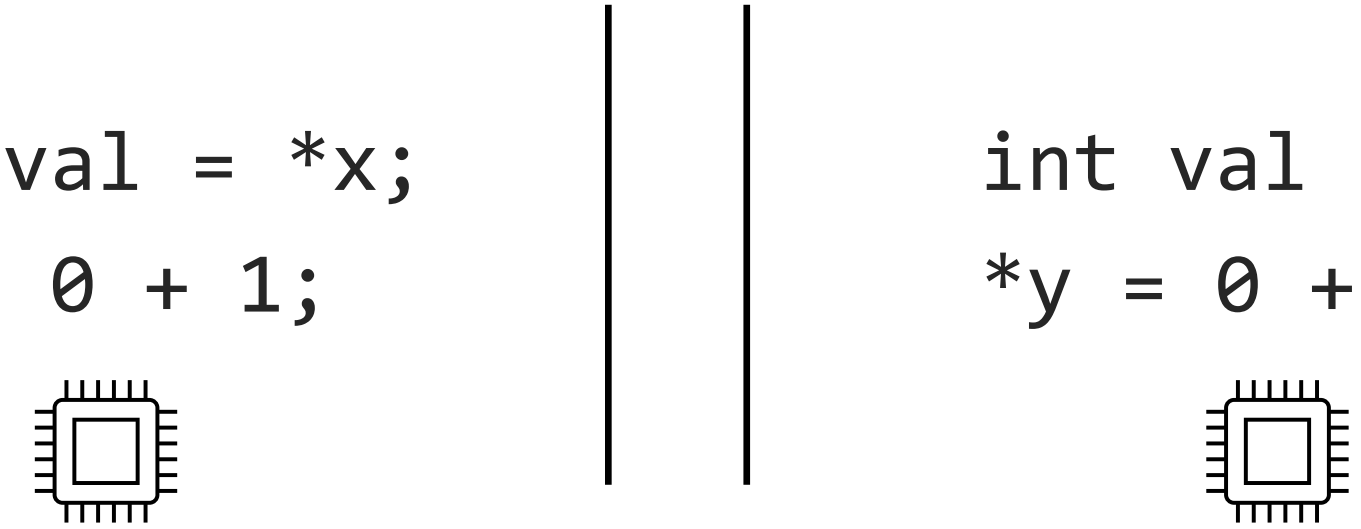
Nobody has responded yet.

Hang tight! Responses are coming in.

Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

```
int val = *y;
*y = 0 + 1;
```



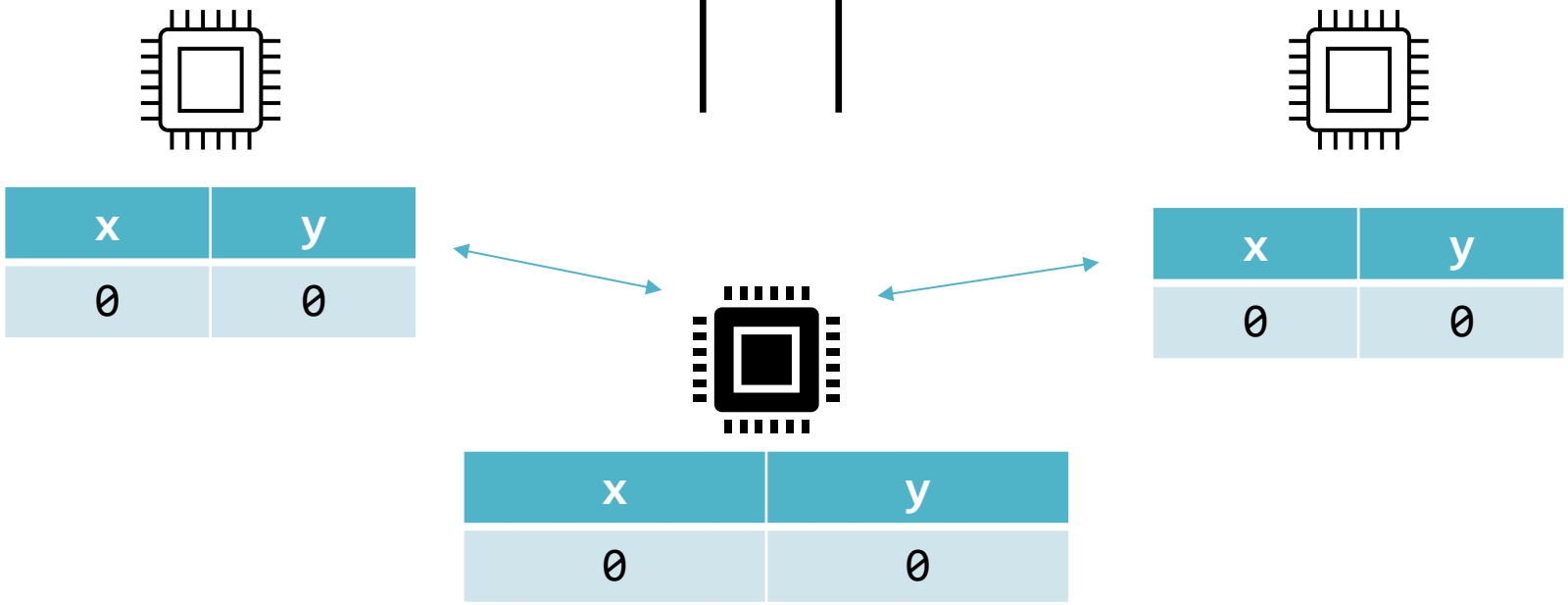
x	y
0	0

Memory

Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

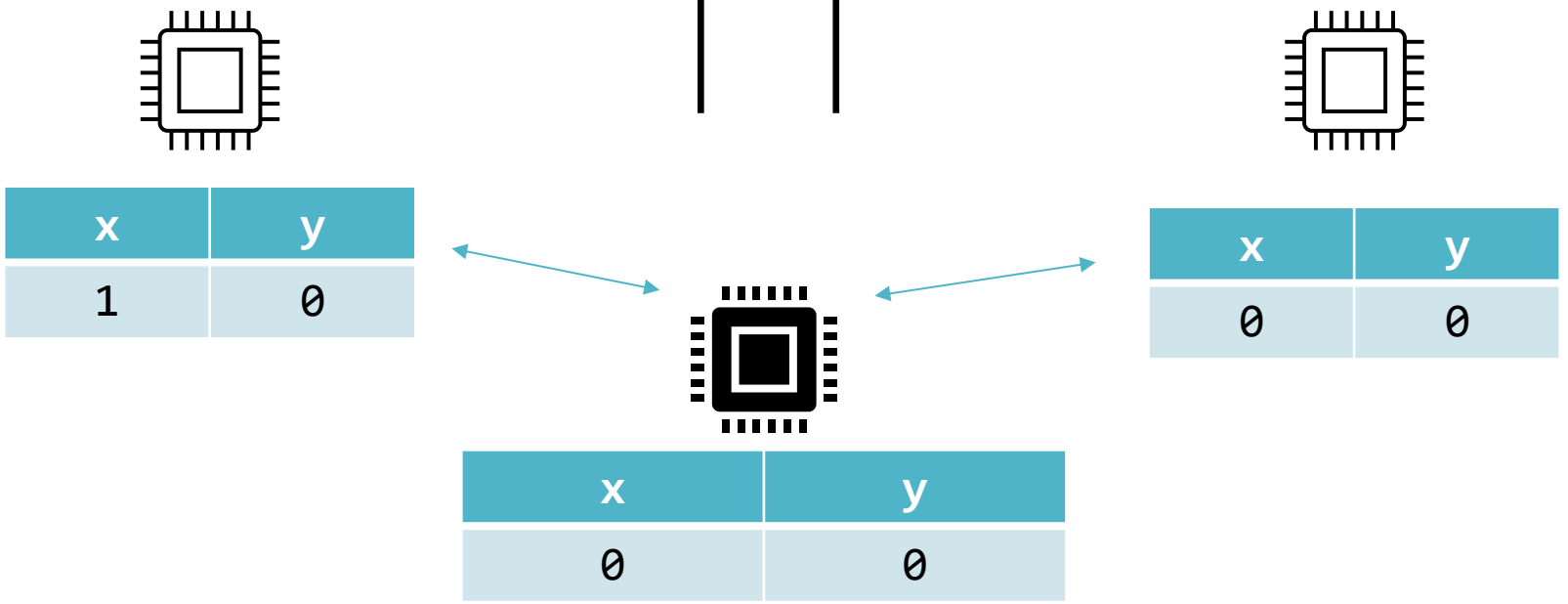
```
int val = *y;
*y = 0 + 1;
```



Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

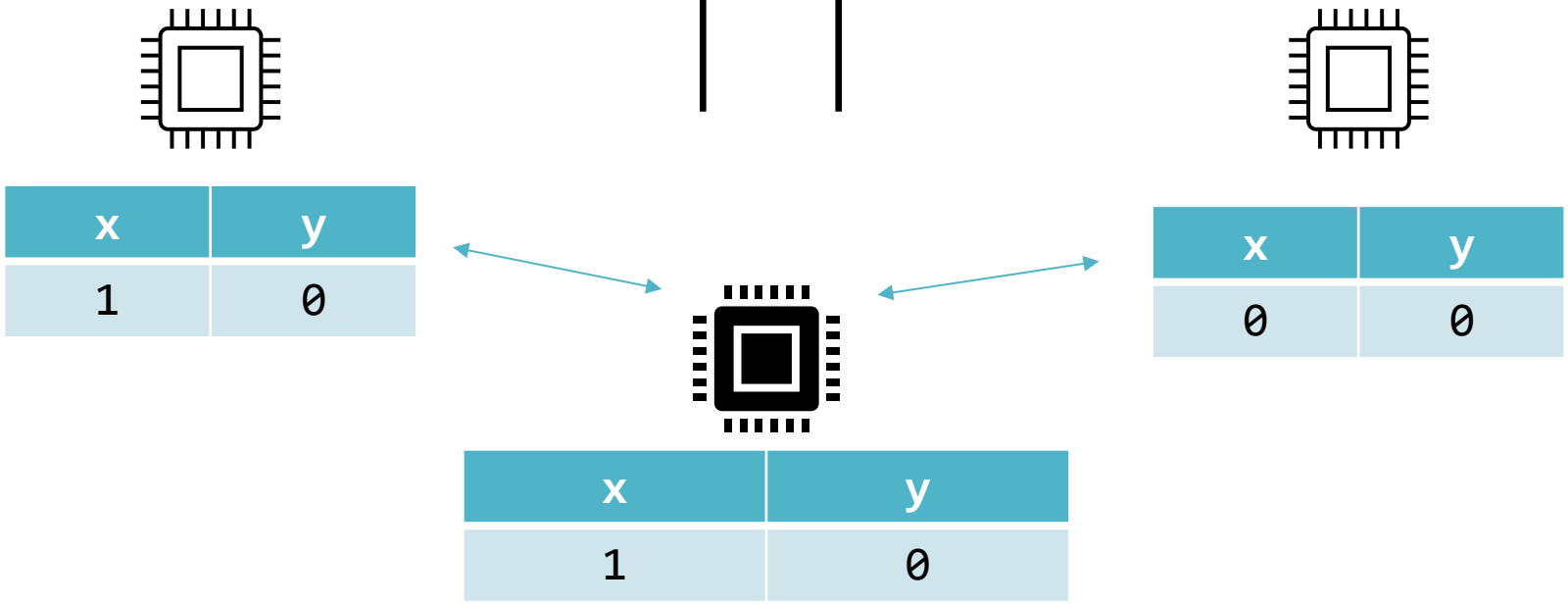
```
int val = *y;
*y = 0 + 1;
```



Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

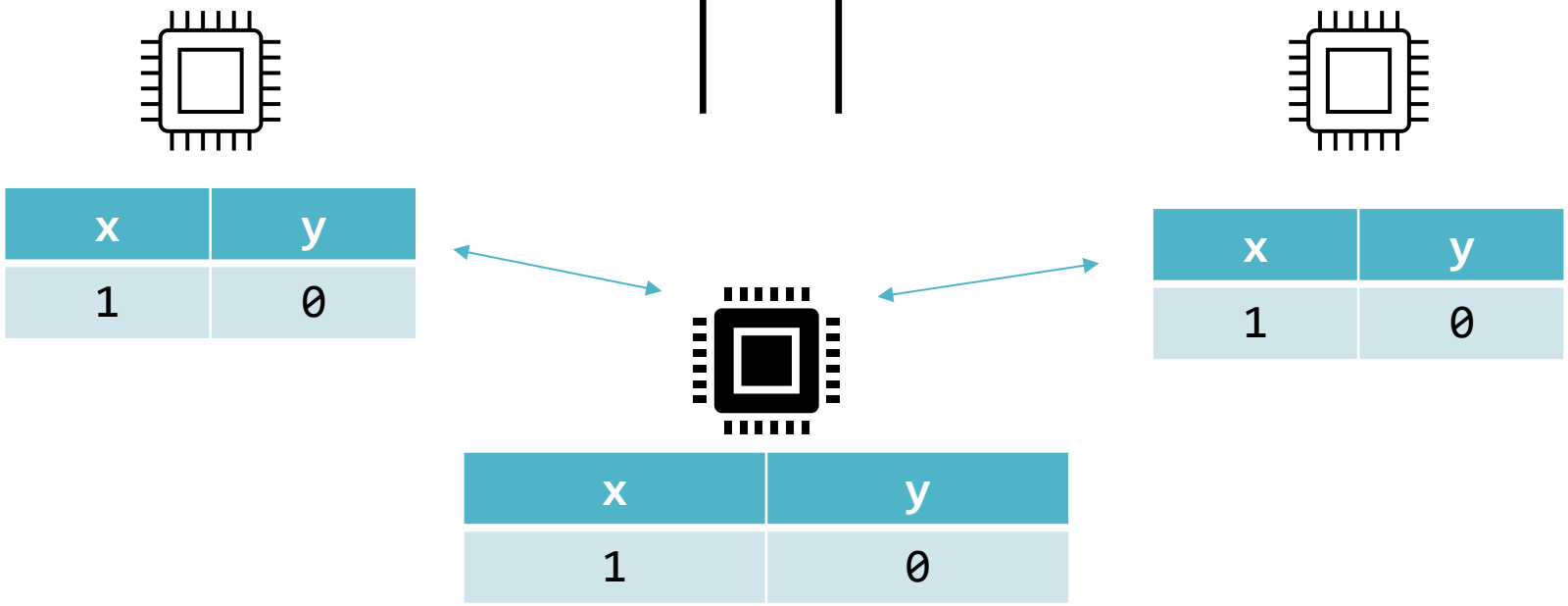
```
int val = *y;
*y = 0 + 1;
```



Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

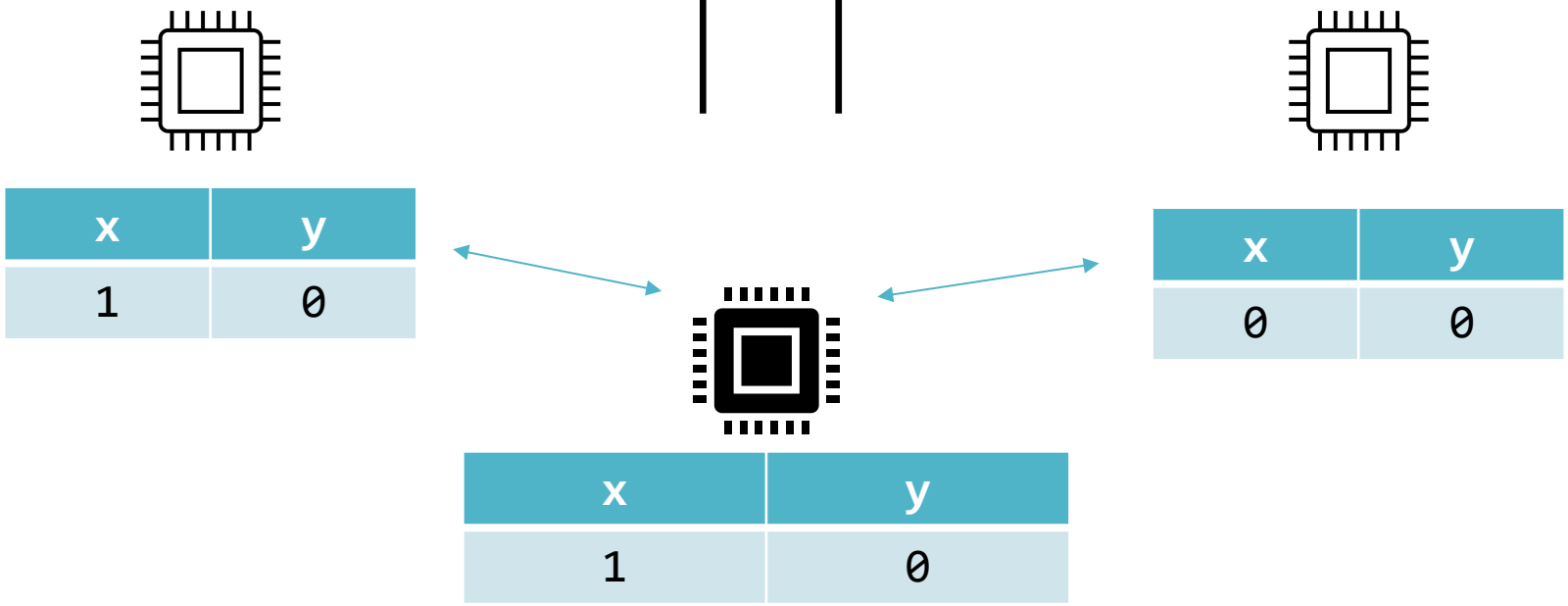
```
int val = *y;
*y = 0 + 1;
```



Concurrent consistency

```
int val = *x;
*x = 0 + 1;
```

```
int val = *y;
int val2 = *x;
```



Cores don’t take turns

- Caches, etc. mean that writes aren’t immediately visible to other threads
- *Different threads have different ideas of what the current value is* at a single location

- So, what *do* we know about the state of memory at any point? And what can threads know about each other’s state?
- The answers come from *memory consistency models*

Memory models

- What can we assume about concurrent memory operations?
  - Will writes happen in order? Will reads happen in order? Will writes to the same location happen in order?
  - What can the programmer do to request stronger guarantees when needed?
  - How do we use just enough synchronization to make a program work?
  - Are some interactions just too dangerous to use at all (data races)?
- This question gets answered twice: at the *hardware* level, and at the *language* level
- This course: understand the different kinds of answers and what we can do with them
- Goal: how do you know you've written a correct concurrent program?

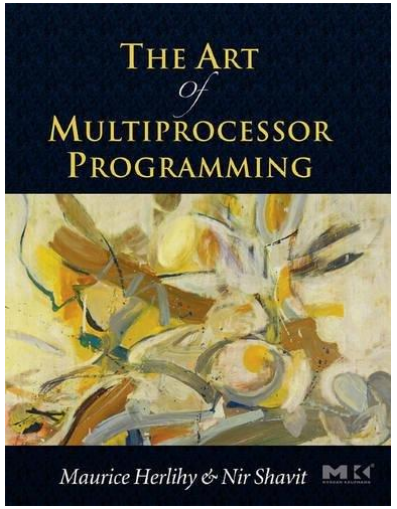
Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Calibration

- Have you:
  - Written a concurrent/multithreaded program? (using fork, pthread, GPU programming, …)
  - Taken CS 454 (Principles of Concurrent Programming) or a similar class?
  - Taken a class where you wrote mathematical proofs?
  - Learned about programming language *semantics* (e.g., small-step and big-step operational semantics)?
  - Mathematically proved the correctness of a program, either on paper or with tools (model checking, static analysis, theorem proving)?



- Submit on Gradescope (entry code J6GYJV), Exercise 8/25

Class Goals and Structure

- What are memory models? How are they expressed? What do they let us do?
- 3 modes: math, code, tools
- Most classes will have in-class exercises
- Exercises are practice for the homework; homework is practice for the final project

Grading

- In-class exercises: 25%
- Assignments: 50%
- Project: 25%

Exercises

- In each class, we'll work through some example problems/proofs
- Submit via Gradescope
- Due at the start of the next class
- You get credit as long as you make some progress on the problem

- Feel free to discuss with your neighbors, ask questions, suggest other approaches, etc.!

Assignments

- Programming/proving assignments
- Submit via Gradescope
- Due at 11:59 PM on the due date
- You can discuss strategy with other students, but don't look at each other's solutions/code!
- Cite your sources (websites, other students, StackOverflow, ChatGPT, etc.)
- You'll get most of the credit for attempting a problem, even if you don't finish it – do what you can, and we'll work through tricky ones in class after the deadline

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.