Weak Memory Concurrency-I

Soham Chakraborty

22.02.2023

Recap

- Concurrency primitives
- Concurrency bugs
- Concurrency analysis

Memory Consistency

Defines the possible result/outcome of a program

Defines the ordering of execution of shared memory accesses

Captures the effect of out-of-order executions and the effects of caches in a processor

Different consistency models:

- Sequential consistency (SC)
- Total-store-order (TSO)
- Partial store order (PSO)
- Relaxed memory order (RMO)

:

Sequential Consistency

"... the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program.“

Leslie Lamport,
"How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs",
IEEE Trans. Comput. C-28,9 (Sept. 1979), 690-691.

How does concurrent programs execute?

"…the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program." ~ Leslie Lamport

Sequential consistency (SC) ==> Interleaving Execution

Representative Example: Store Buffer (SB)

```
            X=Y=0

X = 1;        ||        Y = 1;
a = Y;        ||        b = X;
```

Program: Store Buffer (SB)

X=Y=0

```
S1: X = 1;        ||        S3: Y = 1;
S2: a = Y;        ||        S4: b = X;
```

Outcome:

a=0, b=1 (S1;S2;S3;S4)

Program: Store Buffer (SB)

X=Y=0

S1: X = 1;	S3: Y = 1;
S2: a = Y;	S4: b = X;

Outcome:

a=0, b=1 (S1;S2;S3;S4)
a=1, b=1 (S1;S3;S2:S4 , S3;S1;S4;S2, …)

Program: Store Buffer (SB)

```
            X=Y=0

S1:X = 1;   ||   S3:Y = 1;
S2:a = Y;   ||   S4:b = X;
```

Outcome:

a=0, b=1 (S1;S2;S3;S4)
a=1, b=1 (S1;S3;S2:S4 , S3;S1;S4;S2, …)
a=1, b=0 (S3;S4;S1;S2)
a=0, b=0 **X**

Program: Store Buffer (SB)

X=Y=0

```
X = 1;   ||   Y = 1;
a = Y;   ||   b = X;
```

Let’s execute the program…

Program: Store Buffer (SB)

X=Y=0

X = 1;	‖	Y = 1;
a = Y;	‖	b = X;

Outcome:

a=0, b=1

a=1, b=1

a=1, b=0

a=0, b=0

Program: Store Buffer (SB)

```
        X=Y=0

X = 1;    ||    Y = 1;
a = Y;    ||    b = X;
```

a=0, b=0

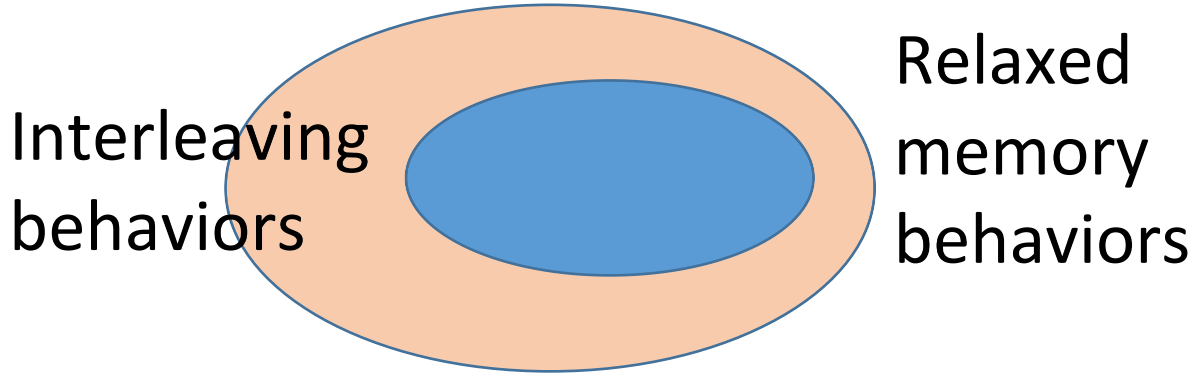
Cannot be explained by thread interleaving.

Relaxed memory model/consistency/concurrency

Relaxed Memory Concurrency

Traditionally: Concurrency = thread interleaving

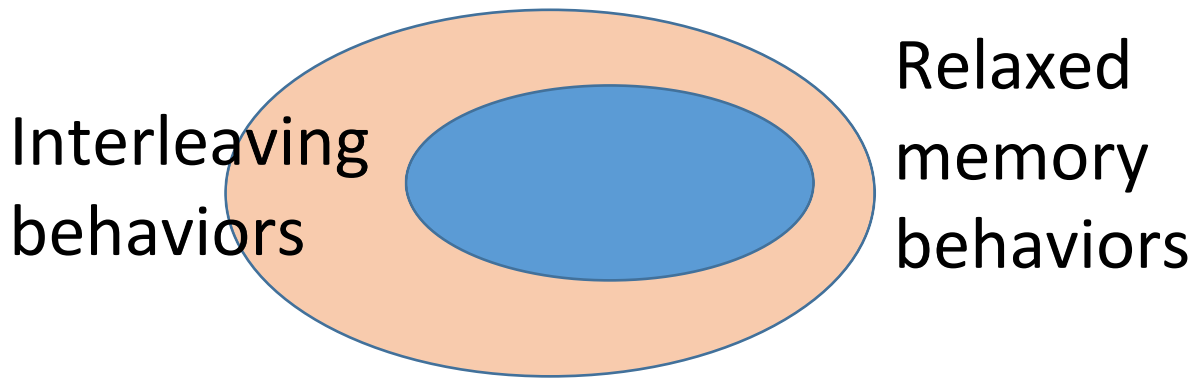
Reality: more behaviors than thread interleaving



Relaxed Memory Concurrency

Traditionally: Concurrency = thread interleaving

Reality: more behaviors than thread interleaving



Reasons:

Compiler: reorder instructions

Hardware:
- Out of order execution
- Data movement/communication is not instantaneous

Order relaxation

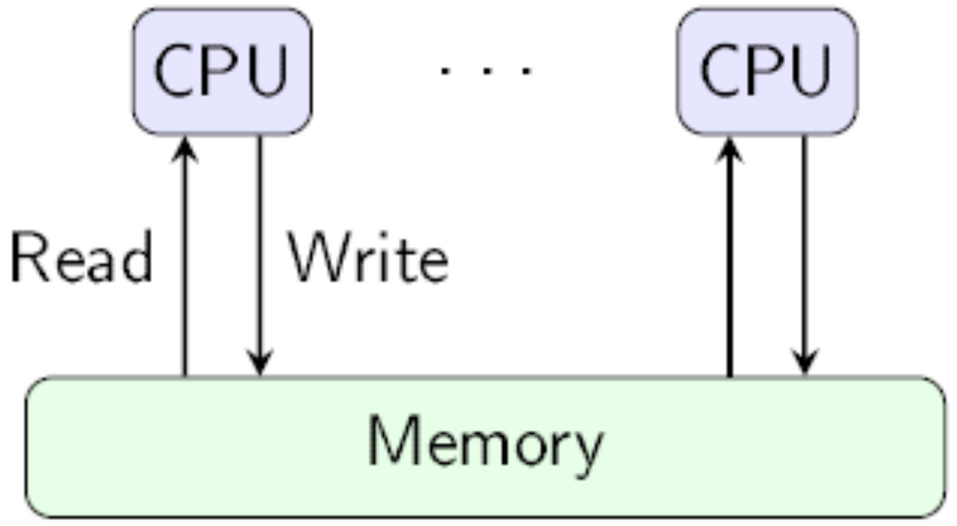
No order for Store – Load

X=Y=0

```
X = 1;          ||          Y = 1;
a = Y;          ||          b = X;
```

a=0, b=0

Memory Models in Processors



Sequential Consistency (SC)

Initially $X = Y = 0$;

$$\begin{array}{l||l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

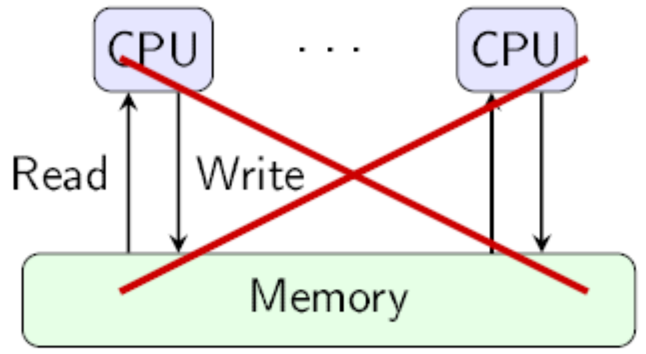
$a = 1, b = 1, \quad X = 1, Y = 1$ ✓

$a = 0, b = 1, \quad X = 1, Y = 1$ ✓

$a = 1, b = 0, \quad X = 1, Y = 1$ ✓

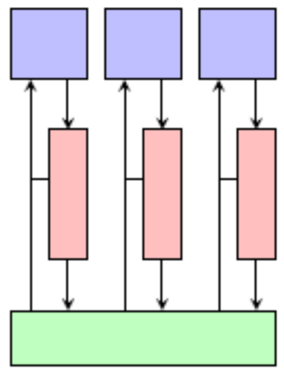
$a = 0, b = 0, \quad X = 1, Y = 1$ ✗

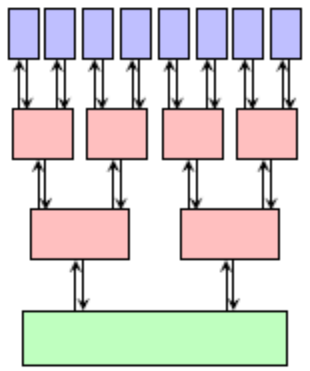
Effect of Caches



Sequential Consistency (SC)

Relaxed Memory Consistency

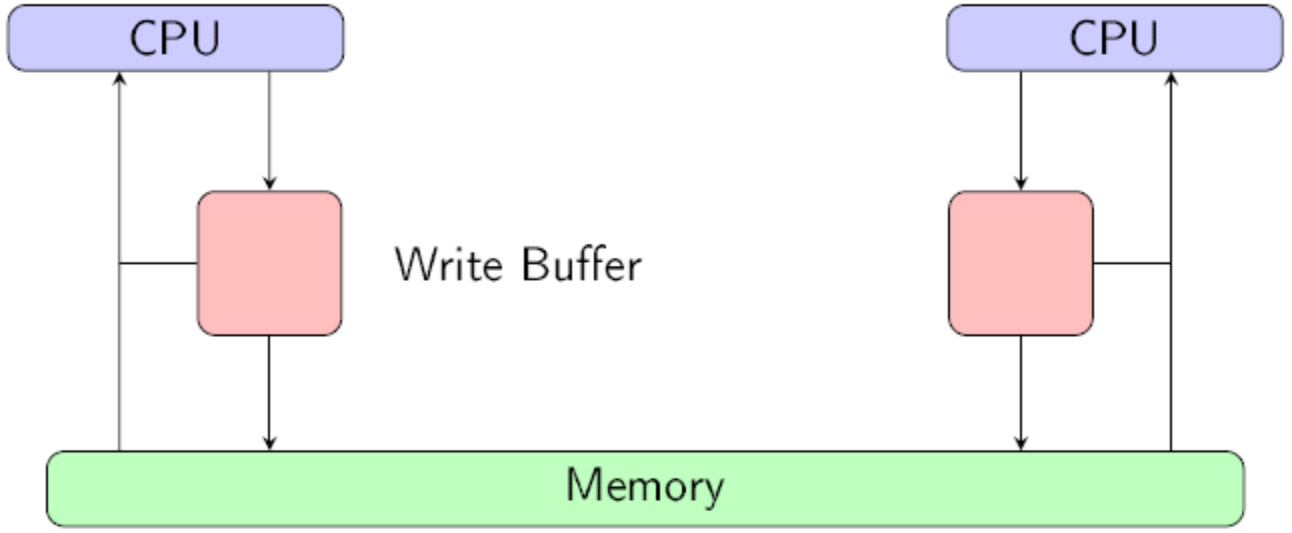




x86 TSO Architecture

Initially $X = Y = 0$;

$$\begin{array}{l|l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

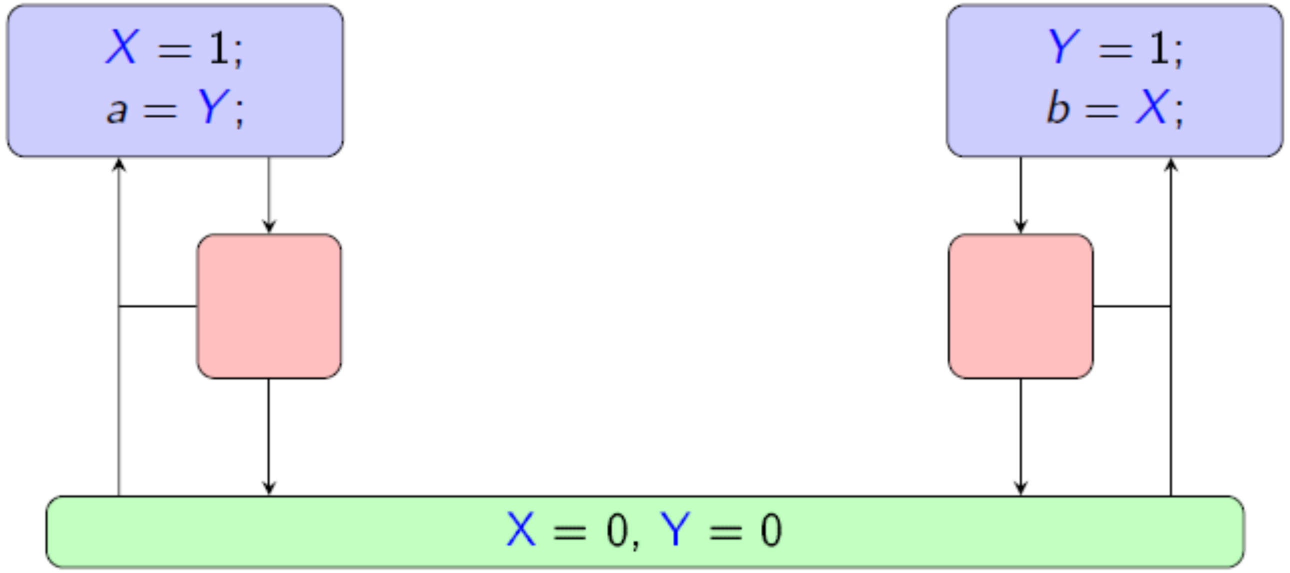


X86 Architecture

x86 TSO Architecture

Initially $X = Y = 0$;

$$\begin{array}{l||l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

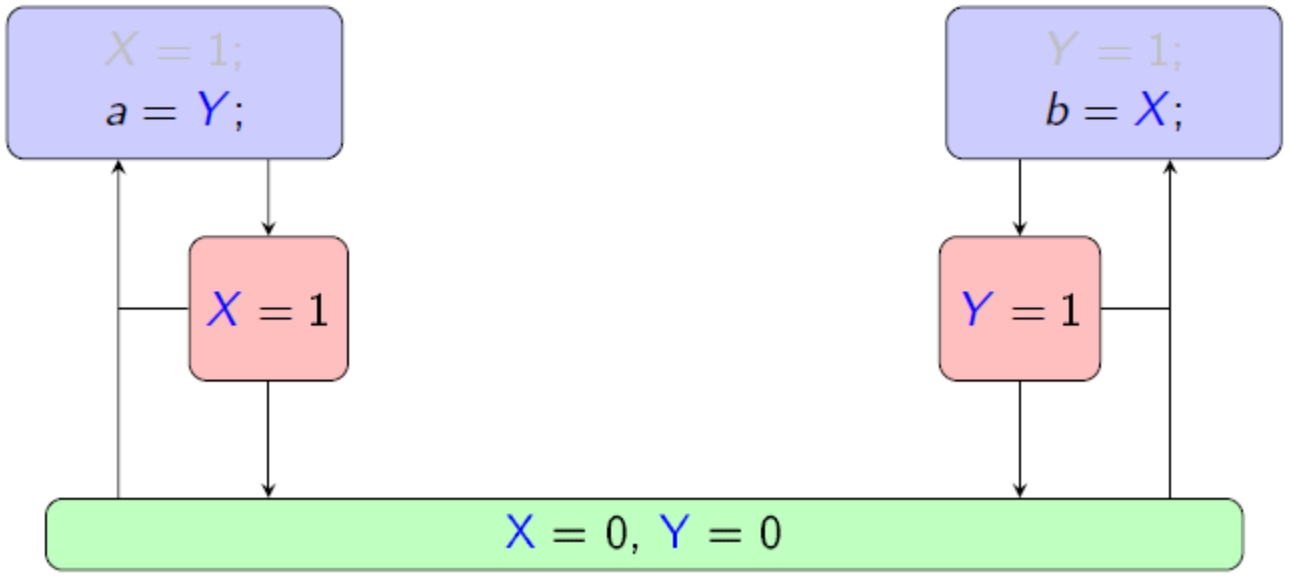


X86 Architecture

x86 TSO Architecture

Initially $X = Y = 0$;

$$\begin{array}{l||l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

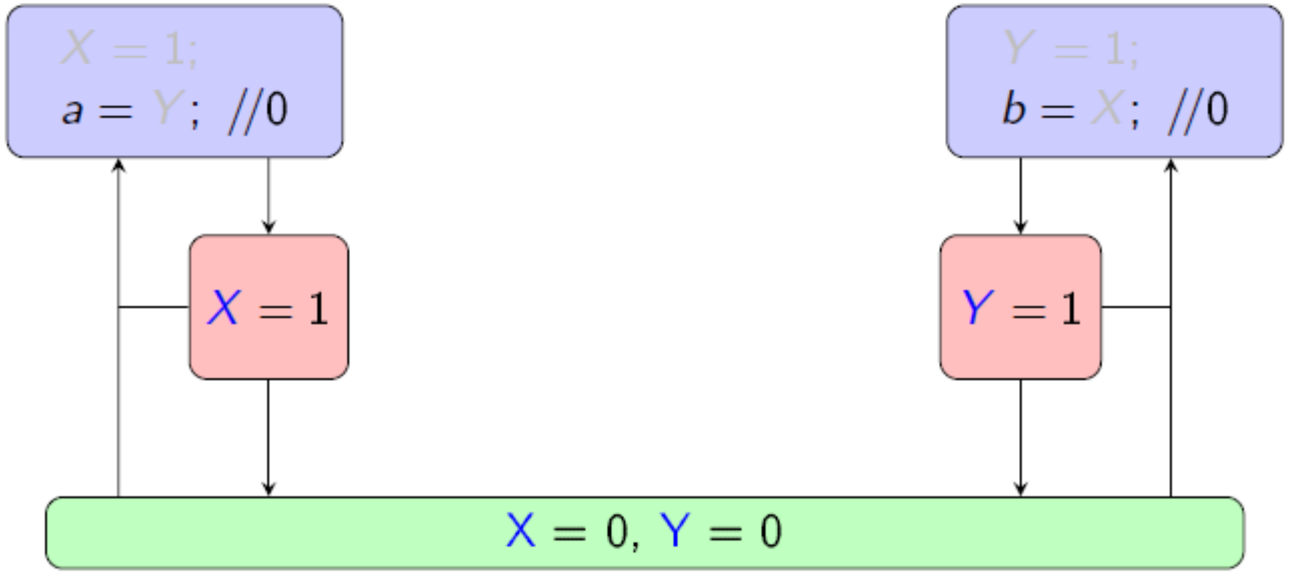


X86 Architecture

x86 TSO Architecture

Initially $X = Y = 0$;

$$\begin{array}{r||l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

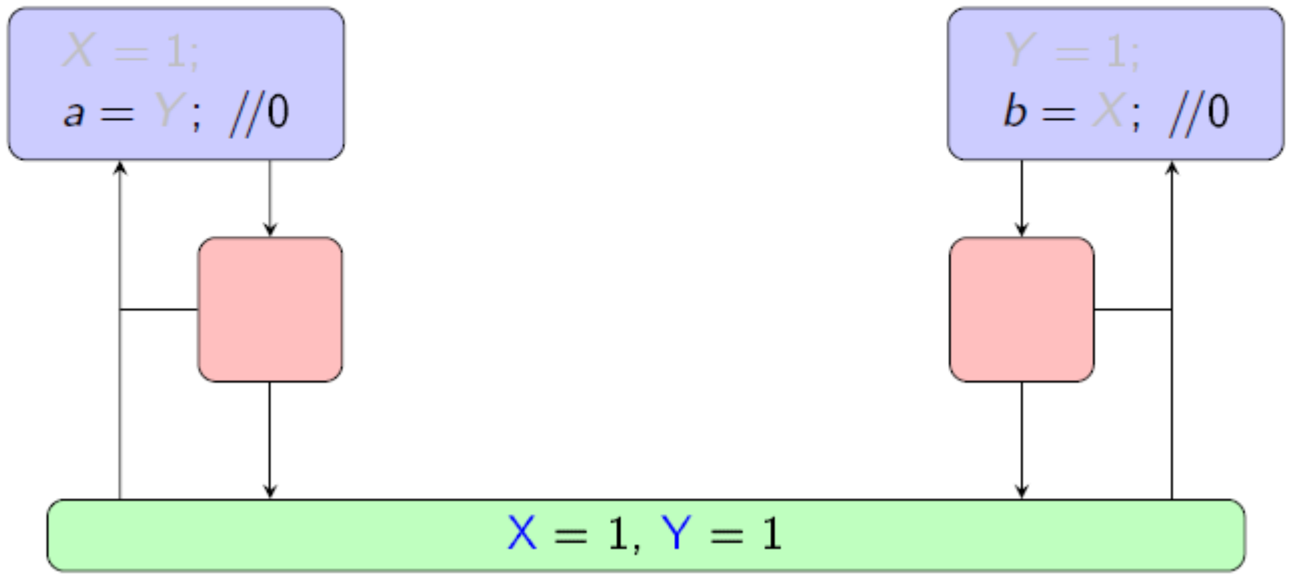


X86 Architecture

x86 TSO Architecture

Initially $X = Y = 0$;

$$\begin{array}{l|l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$



X86 Architecture

x86 TSO Architecture

$$\text{Initially } X = Y = 0;$$

$$\begin{array}{r||l} X = 1; & Y = 1; \\ a = Y; & b = X; \end{array}$$

$$\begin{array}{ll} a = 1, b = 1, \quad X = 1, Y = 1 & \checkmark \\ a = 0, b = 1, \quad X = 1, Y = 1 & \checkmark \\ a = 1, b = 0, \quad X = 1, Y = 1 & \checkmark \\ a = 0, b = 0, \quad X = 1, Y = 1 & \checkmark \end{array}$$

X86 Architecture

Total Store Order (TSO)

Relaxation for Store – Load

a;b	Store	Load
Store	N	Y
Load	N	N

Behavior = transformations+interleaving

Another Program: Message Passing (MP)

```
        X=Y=0

S1: X = 1;   ||   S3: a = Y;
S2: Y = 1;   ||   S4: if(a == 1)
                  S5:    b = X;
```

Is a=1, b=0 possible an allowed behavior?

- SC ?
- TSO ?

S3;S4;S1;S2 ➔ a=0, b= …

S1;S2;S3;S4;S5 ➔ a=1, b=1

…

Another Program: Message Passing (MP)

X=Y=0

```
S1: X = 1;        ||        S3: a = Y;
S2: Y = 1;        ||        S4: if(a == 1)
                            S5:    b = X;
```

Is a=1, b=0 possible an allowed behavior?

- SC ?
- TSO ?

TSO has same behaviors as SC model

Another Program: Message Passing (MP)

X=Y=0

```
X = 1;         ||    a = Y;
Y = 1;         ||    if(a == 1)
                        b = X;
```

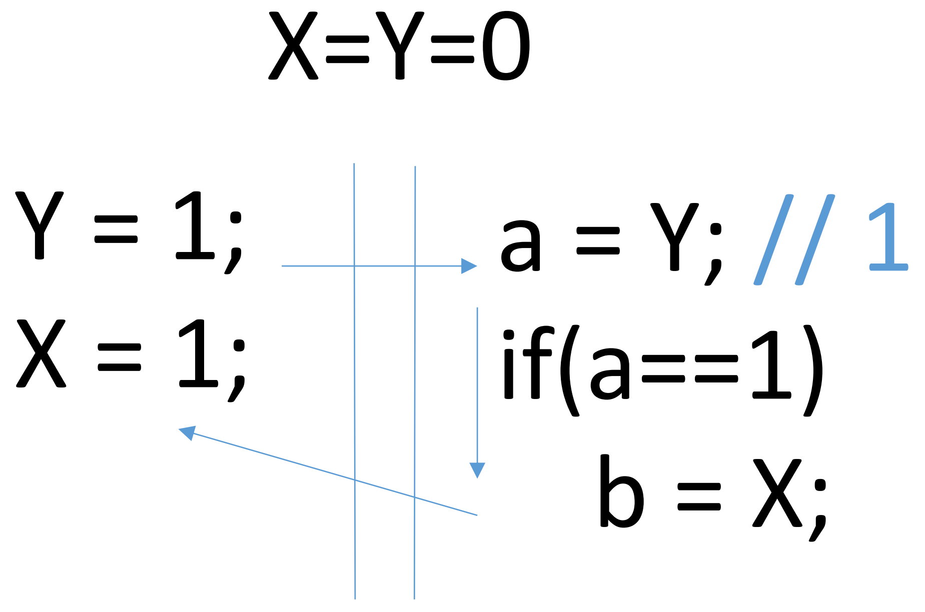
Is a=1, b=0 possible an allowed behavior?

- SC ***X***
- TSO ***X***

Program: Message Passing (MP)

X=Y=0

```
X = 1;   ||  a = Y;
Y = 1;   ||  if(a==1)
         ||      b = X;
```



X=Y=0

```
Y = 1;   ||  a = Y; // 1
X = 1;   ||  if(a==1)
         ||      b = X;
```

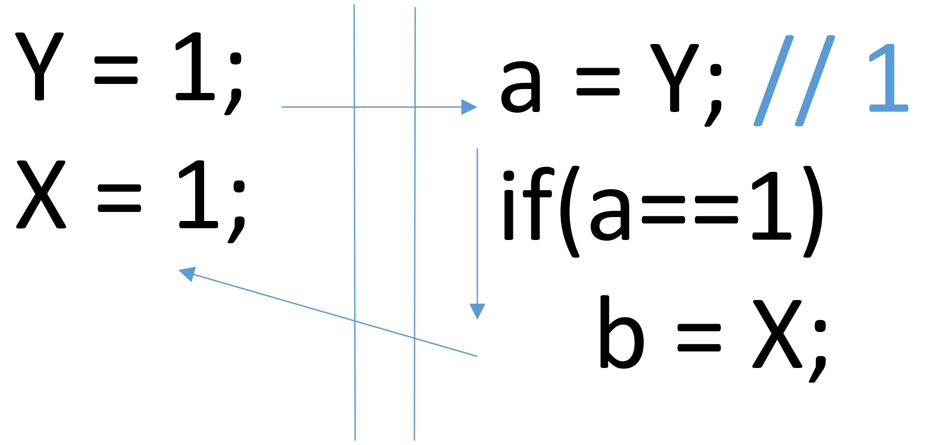
a =1, b = 0

Requires Store-Store relaxation

Program: Message Passing (MP)

```
X=Y=0

X = 1;    ||    a = Y;
Y = 1;    ||    if(a==1)
                    b = X;
```



```
X=Y=0

Y = 1;    ||    a = Y; // 1
X = 1;    ||    if(a==1)
                    b = X;
```

a =1, b = 0

PSO model allows Store-Store relaxation

Partial Store Order (PSO)

Relaxation for Store – Store, Store - Load

a;b	Store	Load
Store	Y	Y
Load	N	N

Another Program: Message Passing (MP)

```
X=Y=0

X = 1;     ||     a = Y;
Y = 1;     ||     if(a == 1)
                     b = X;
```

Is a=1, b=0 possible an allowed behavior?

- SC ***no***
- TSO ***no***
- PSO ***yes***

Yet Another Program: Load Buffering (LB)

X=Y=0

S1: a = X;	S3: b = Y;
S2: Y = 1;	S4: X= 1;

Is a=1, b=1 possible an allowed behavior?

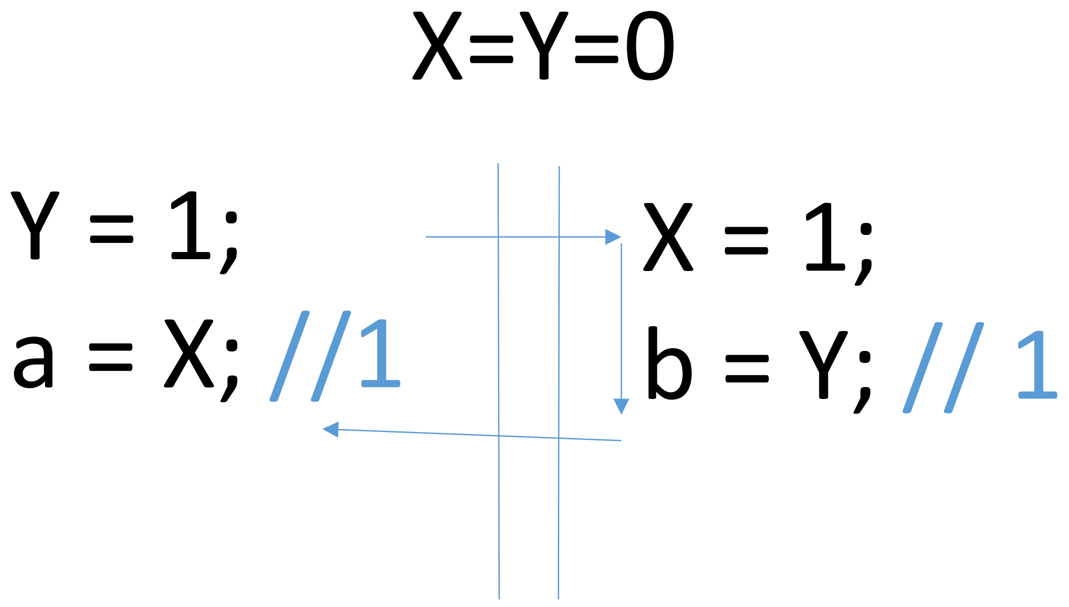
- SC ? X
- TSO ? X
- PSO ? X

a=1, b=0 ➔ S3;S4
a=0,b=0 ➔ S1;S3; …
a=0, b=1 ➔

Program: Message Passing (MP)

```
        X=Y=0

a = X;   ||   b = Y;
Y = 1;   ||   X=1;
```



```
        X=Y=0

Y = 1;        ||   X = 1;
a = X; //1    ||   b = Y; // 1
```

a =1, b = 1

Requires Load-Store relaxation

Relaxed Memory Order (RMO)

Relaxation for Store – Store

a;b	**Store**	**Load**
Store	Y	Y
Load	Y	Y

Summary

ACC. Pair Relaxation → / Memory Model ↓	Store-Load	Store-Store	Load-Load	Load-Store
SC	N	N	N	N
TSO	Y	N	N	N
PSO	Y	Y	N	N
RMO	Y	Y	Y	Y

Challenges in Weak Memory Concurrency

* How do we understand program behaviors under weak memory?

Formal semantics of concurrency models

- Transformation + interleaving
- Operational model
- Axiomatic model

* Which program transformations are correct?

Note: reordering accesses is an incorrect transformation under SC

* How do we verify the propgram properties?

Traditional tools analyze interleavings, requires new techniques.

Other Primitives & Properties

- Dependencies: enforce ordering in Arm, Power architectures
- Fences: order memory accesses
- Atomic update: compare-and-swap, compare-exchange, atomic- increment, …
- Coherence: same-location accesses are ordered
- Multicopy-atomicity: visibility of a write to the other threads

References

Chapter 5.6 (Memory consistency Models)

- Computer Architecture, Sixth Edition A Quantitative Approach by John L. Hennessy, David A. Patterson.

Chapter 2. Correct Compilation of Relaxed Memory Concurrency.
PhD Thesis. 2019.
Soham Chakraborty.
(Link)