Memory consistency models and synchronizations

Acknowledgements

- Several of the slides in the deck are from Luis Ceze (Washington), Nima Horanmand (Stony Brook), Mark Hill, David Wood, Karu Sankaralingam (Wisconsin), Abhishek Bhattacharjee(Rutgers).

- Development of this course is partially supported by Western Digital corporations.

Coherence vs. Consistency

```
                A=0   FLAG=0
C 0                            C 1
ST A 1;                        L1: LD r1 FLAG
ST FLAG 1;                     If  r1 == 0; JMP L1;// spin lock
                                  LD r2 A;
```

Coherence vs. Consistency

```
                    A=0   FLAG=0
C_0                                 C_1
ST A 1;                             L1: LD r1 FLAG
ST FLAG 1;                          If  r1 == 0; JMP L1;// spin lock
                                       LD r2 A;
```

What value should r2 contain?

Coherence vs. Consistency

```
                  A=0   FLAG=0
      C 0                          C 1
      ST A 1;                      L1: LD r1 FLAG
      ST FLAG 1;                   If  r1 == 0; JMP L1;// spin lock
                                     LD r2 A;
```

What value should r2 contain? (expect r2=1)

But what does the coherence says?

Coherence vs. Consistency

```
                A=0   FLAG=0
    C 0                         C 1
    ST A 1;                     L1: LD r1 FLAG
    ST FLAG 1;                  If  r1 == 0; JMP L1;// spin lock
                                  LD r2 A;
```

What value should r2 contain? (expect r2=1)

But what does the coherence says? Nothing

Coherence is about total ordering of stores and loads to **one** given memory location (in practice a single cache block)

Says nothing about ordering across different memory locations/addresses

Coherence vs. Consistency

```
                 A=0   FLAG=0
   C 0                           C 1
   ST A 1;                       L1: LD r1 FLAG
   ST FLAG 1;                    If  r1 == 0; JMP L1;// spin lock
                                   LD r2 A;
```

What value should r2 contain? (expect r2=1)

But what does the coherence says? Nothing

Coherence is about total ordering of stores and loads to **one** given memory location (in practice a single cache block)

Says nothing about ordering across different memory locations/addresses

But correctly specifying/writing multi-threaded program requires guarantees for ordering of load/stores to different locations

Memory consistency models

A memory consistency model specifies global orders of writes to **all memory** locations relative to each other

- A memory consistency model tells what are the legal reordering of loads/stores to different memory locations

- Different memory consistency models possible
  - Tradeoff between ease of programmability vs. performance
  - "Relaxed" models enforces less ordering (better performance) but harder to program

- Memory consistency model part of ISA
  - X86 and ARM has different memory consistency model

Memory consistency models

- Specifies which re-ordering of loads and stores (to different addresses) are allowed

```
                A=0   FLAG=0
C 0                         C 1
ST A 1;                     L1: LD r1 FLAG
ST FLAG 1;                  If  r1 == 0; JMP L1;// spin lock
                               LD r2 A;
```

Memory consistency models

- Specifies which re-ordering of loads and stores (to different addresses) are allowed

```
                 A=0   FLAG=0
C 0                          C 1
ST A 1;                      L1: LD r1 FLAG
ST FLAG 1;                   If  r1 == 0; JMP L1;// spin lock
                               LD r2 A;
```

- For example, this program will work fine if load, store bypassing is not allowed within a given thread, even if they are to different addresses

Memory consistency models

- Specifies which re-ordering of loads and stores (to different addresses) are allowed

```
                A=0   FLAG=0
C_0                             C_1
ST A 1;                         L1: LD r1 FLAG
ST FLAG 1;                      If  r1 == 0; JMP L1;// spin lock
                                   LD r2 A;
```

- For example, this program will work fine if load, store bypassing is not allowed within a given thread, even if they are to different addresses
- Parallel programmers rely on the memory model to reason about correctness of your program

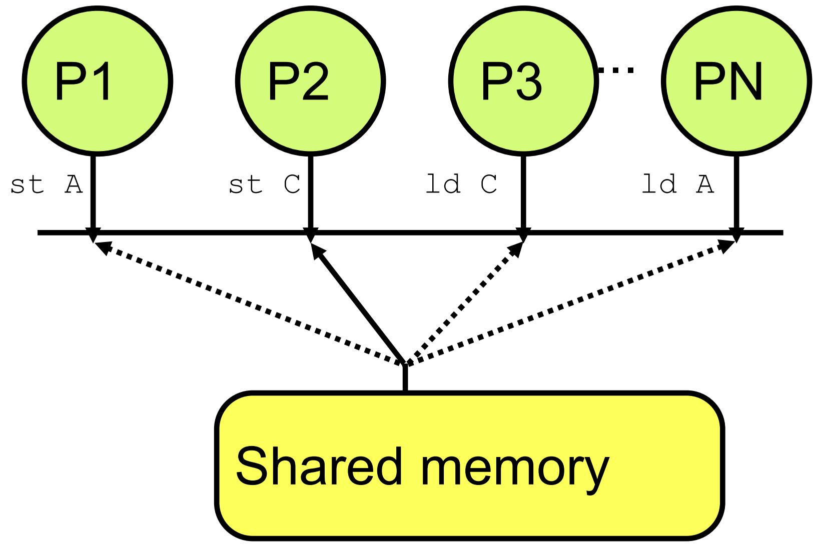
Who defines memory consistency models?

- Any programmer of parallel shared memory programs should care of memory consistency models
- Defined by the H/W –described in ISA
- Compilers should also need to define their memory consistency models
  - Determines what optimizations are allowed

Many possible memory consistency models

- Sequential consistency
  - Most intuitive, programmer friendly
  - But most restriction in terms of performance optimizations
  - e.g., implemented in MIPS R10K
- Total store order (processor consistency model)
  - Less restrictive than
  - Intel, AMD's x86'64 processors
  - Sun UltraSPARC
- Weak memory models (release consistency models)
  - Most relaxed, burdens programmer but allows more hardware optimizations
  - ARM, IBM POWER processors

Sequential consistency

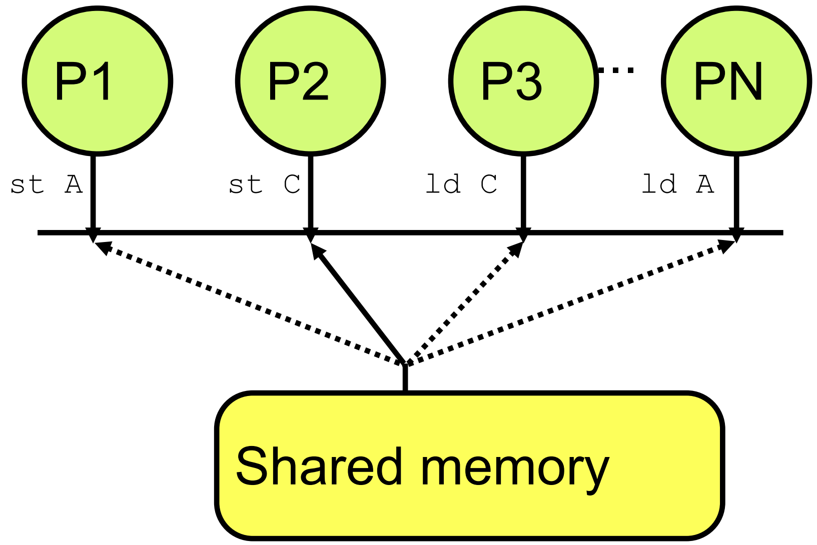


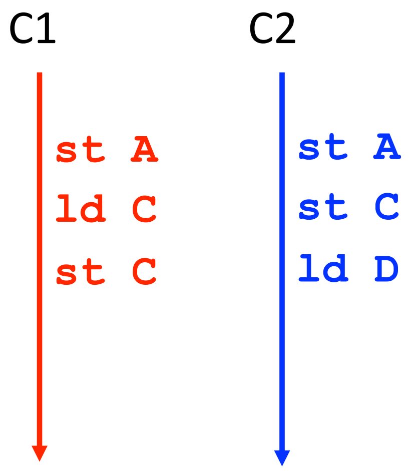
Per-processor program order: memory operations from individual processors maintain program order

Single sequential order: the memory operations from all processors maintain a single sequential order

[Lamport'79]

Sequential consistency



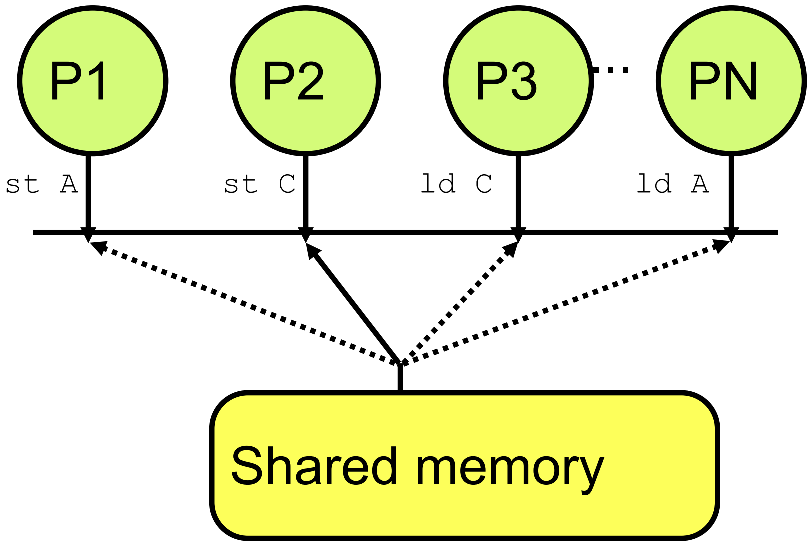


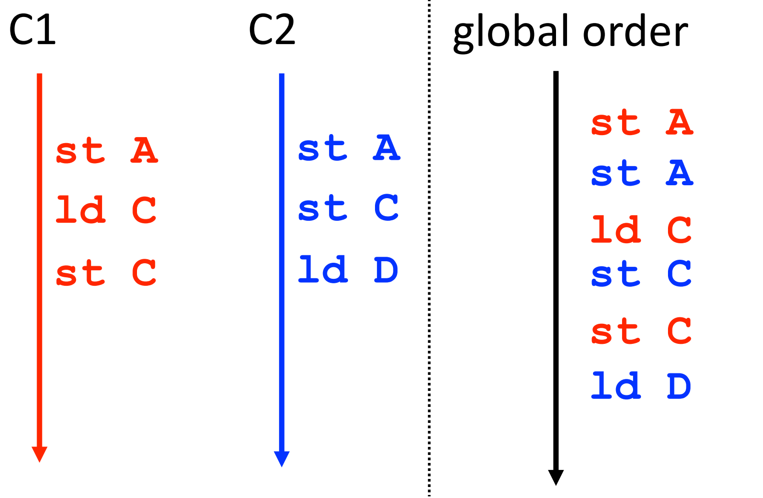
C1	C2
st A	st A
ld C	st C
st C	ld D

Per-processor program order: memory operations from individual processors maintain program order

Single sequential order (Total order): memory operations from all processors maintain a single sequential order

Sequential consistency

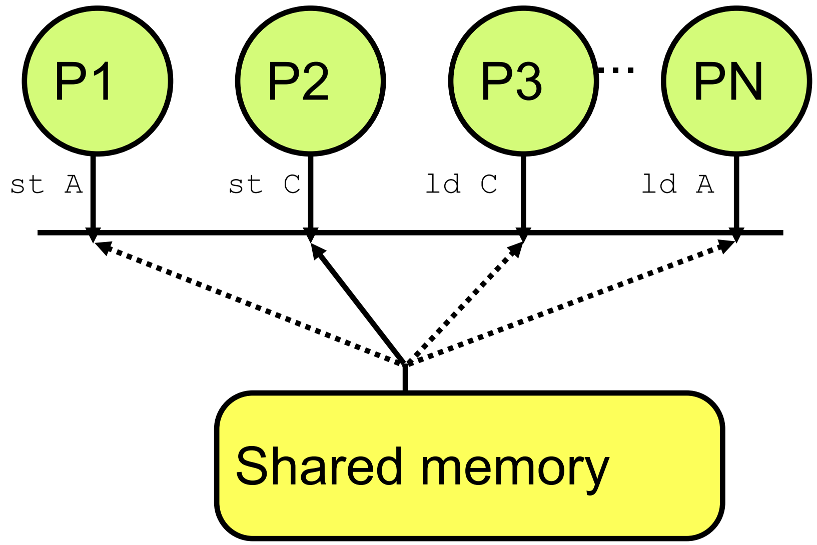


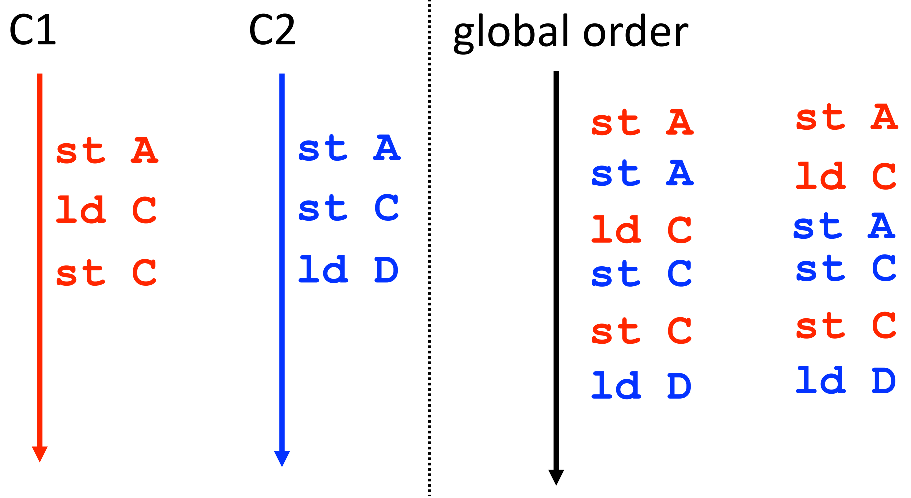


Per-processor program order: memory operations from individual processors maintain program order

Single sequential order (Total order): memory operations from all processors maintain a single sequential order

Sequential consistency





Per-processor program order: memory operations from individual processors maintain program order

Single sequential order (Total order): memory operations from all processors maintain a single sequential order

Sequential consistency (SC)

```
                  A=0   FLAG=0
C_0                            C_1
ST A 1;                        L1: LD r1 FLAG
ST FLAG 1;                     If  r1 == 0; JMP L1;// spin lock
                                  LD r2 A;
```

- What will be the value loaded in r2 under SC?

Reordering of load/stores

Program order

Earlier operation / Later operation	LD	ST
LD	NO	NO
ST	NO	NO

Allowed reordering of load/stores to different addresses under sequential consistency.

No reordering for load/stores to same address – ensured by coherence

Food for thought (assume SC)

- Answer the following questions:
  - Initially: all variables zero (that is, x is 0, y is 0, flag is 0, A is 0)
  - What value pairs can be read by the two loads? (x, y) pairs:

Core 0	Core 1

`LD x`
`LD y`
	`ST y 1`
	`ST x 1`