Dependence Analysis

Motivating question

- Can the loops on the right be run in parallel?
  - *i.e.*, can different processors run different iterations in parallel?
- What needs to be true for a loop to be parallelizable?
  - Iterations cannot interfere with each other
  - No *dependence* between iterations

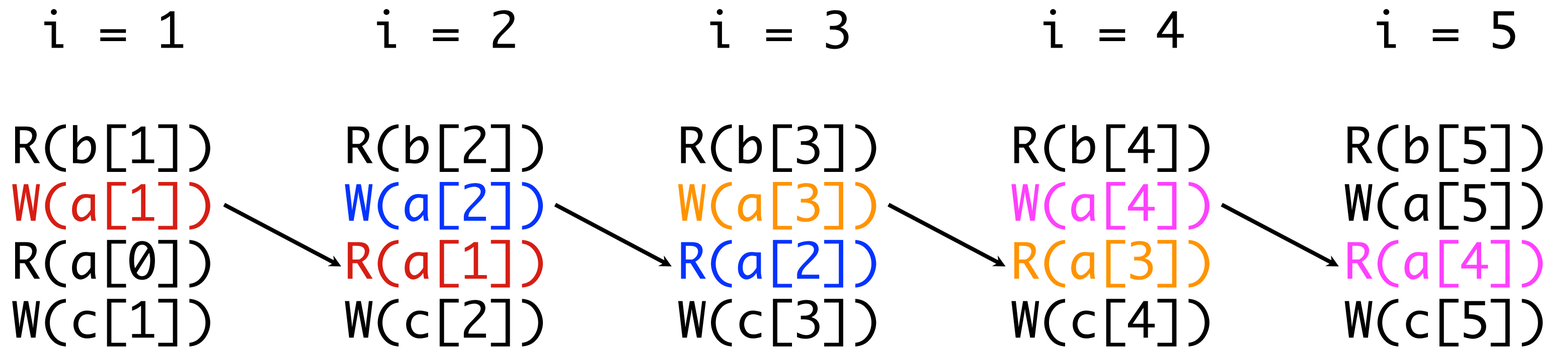
```
for (i = 1; i < N; i++) {
 a[i] = b[i];
 c[i] = a[i - 1];
}
```

```
for (i = 1; i < N; i++) {
  a[i] = b[i];
  c[i] = a[i] + b[i - 1];
}
```

Dependences

```
for (i = 1; i < N; i++) {
 a[i] = b[i];
 c[i] = a[i - 1];
}
```

- A *flow dependence* occurs when one iteration writes a location that a *later* iteration reads



Running a loop in parallel

- If there is a dependence in a loop, we cannot guarantee that the loop will run correctly in parallel
  - What if the iterations run out of order?
    - Might read from a location before the correct value was written to it
  - What if the iterations do not run in lock-step?
    - Same problem!

Other kinds of dependence

- ***Anti dependence*** – When an iteration *reads* a location that a later iteration *writes* (why is this a problem?)

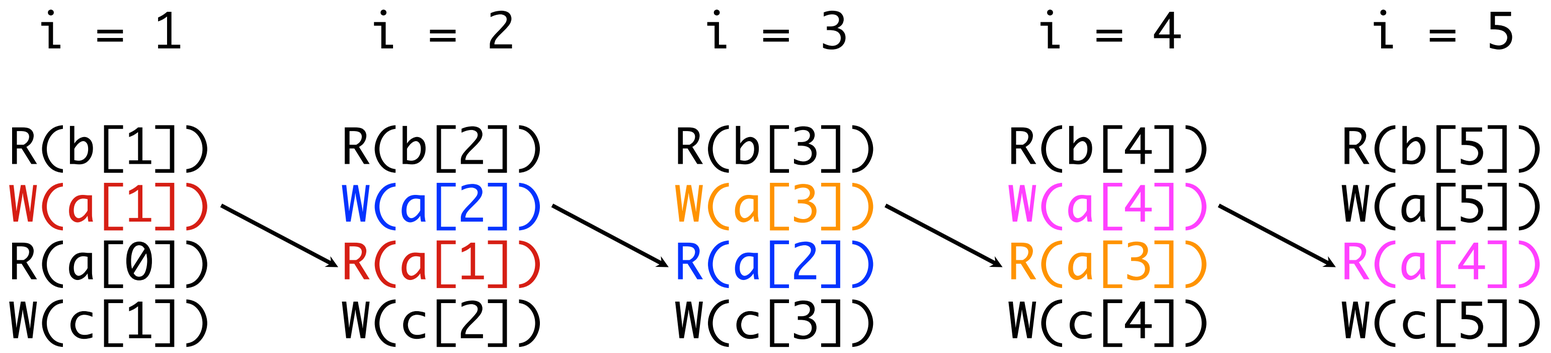
```
for (i = 1; i < N; i++) {
 a[i - 1] = b[i];
 c[i] = a[i];
}
```

- ***Output dependence*** – When an iteration *writes* a location that a later iteration *writes* (why is this a problem?)

```
for (i = 1; i < N; i++) {
 a[i] = b[i];
 a[i + 1] = c[i];
}
```

Data dependence concepts

- Dependence ***source*** is the earlier statement (the statement at the tail of the dependence arrow)
- Dependence ***sink*** is the later statement (the statement at the head of the dependence arrow)



- *Dependences can only go forward in time*: always from an earlier iteration to a later iteration.

Using dependences

- If there are no dependences, we can parallelize a loop
  - None of the iterations interfere with each other
- Can also use dependence information to drive other optimizations
  - Loop interchange
  - Loop fusion
  - How do we represent dependences in loops?