Announcements

Midterm Exam

• Due now

Monday, 18 July

• No lecture

HW2

• Will be made available Wednesday
• Similar in structure to HW1; will include book “questions” and a programming task.
From last time

<table>
<thead>
<tr>
<th>“worst” case</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>“best” case</td>
<td>17</td>
<td>16</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>“average” case</td>
<td>2</td>
<td>5</td>
<td>17</td>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>
From last time

“worst” case

1  2  5  16  17

“best” case

17  16  5  2  1

“average” case

2  5  17  1  16

int m = 0;
for [i = 0 to n-1]
    if (a[i] > m)
        m = a[i];

int m = 0;
co [i = 0 to n-1] {  
    if (a[i] > m) 
        m = a[i];}
From last time

Q: How many checks (whether a[i] is greater than m) are performed in the worst, best, and average cases?

```
int m = 0;
col [i = 0 to n-1] {
    if (a[i] > m)
        if (a[i] > m)
            m = a[i];
}
```
From last time

Q: How many checks (whether a[i] is greater than m) are performed in the worst, best, and average cases?

<table>
<thead>
<tr>
<th>Worst</th>
<th>Best</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-10</td>
<td>5-10</td>
<td>5-10</td>
</tr>
</tbody>
</table>

Q: What is the reason for the ranges?

```
int m = 0;
for (i = 0 to n-1) {
    if (a[i] > m)
        if (a[i] > m)
            m = a[i];
}
```
From last time

“worst” case
1 2 5 16 17

“best” case
17 16 5 2 1

“average” case
2 5 17 1 16

Q: How many checks (whether \(a[i]\) is greater than \(m\)) are performed in the worst, best, and average cases?

Worst: 5-10
Best: 5-10
Average: 5-10

Q: How many updates (to \(m\)) are performed in the worst, best, and average cases?

```c
int m = 0;
c for (i = 0 to n-1) {
    if (a[i] > m)
        m = a[i];
}
```
From last time

| “worst” case | 1 | 2 | 5 | 16 | 17 |
| “best” case  | 17 | 16 | 5 | 2 | 1 |
| “average” case | 2 | 5 | 17 | 1 | 16 |

Q: How many updates (to m) are performed in the worst, best, and average cases?

- Worst: 1-5
- Best: 1-5
- Average: 1-5

Q: How many checks (whether a[i] is greater than m) are performed in the worst, best, and average cases?

- Worst: 5-10
- Best: 5-10
- Average: 5-10

Q: Under what circumstances is the 1 realized? The 5?

```c
int m = 0;
for (i = 0 to n-1) {
    if (a[i] > m) {
        m = a[i];
    }
}
```
From last time

Q: How many checks (whether a[i] is greater than m) are performed in the worst, best, and average cases?

- Worst: 5-10
- Best: 5-10
- Average: 5-10

Q: How many updates (to m) are performed in the worst, best, and average cases?

- Worst: 1-5
- Best: 1-5
- Average: 1-5

Q: What is the total time needed to process each of the worst (W), best (B), and (A) average case arrays, assuming 3 and 5 units of time for check and update?

```c
int m = 0;
co [i = 0 to n-1] {
    if (a[i] > m)
        (if (a[i] > m)
            m = a[i];
        }
```
From last time

"worst" case
1 2 5 16 17

"best" case
17 16 5 2 1

"average" case
2 5 17 1 16

Q: How many checks (whether a[i] is greater than m) are performed in the worst, best, and average cases?

Worst: 5-10
Best: 5-10
Average: 5-10

Time: 15-30

Q: How many updates (to m) are performed in the worst, best, and average cases?

Worst: 1-5
Best: 1-5
Average: 1-5

Time: 5-25

Q: What is the total time needed to process each of the worst (W), best (B), and average (A) case arrays, assuming 3 and 5 units of time for check and update?

W: 20-55
B: 20-55
A: 20-55

int m = 0;
co [i = 0 to n-1] {
    if (a[i] > m)
        (if (a[i] > m)
            m = a[i];)
}
From last time

Q: What is the difference between these two approaches of concurrently finding the largest entry the array a?

int m = 0;
co [i = 0 to n-1] {
  if (a[i] > m)
    m = a[i];
}

int m = 0;
co [i = 0 to n-1] {
  if (a[i] > m)
    if (a[i] > m)
      m = a[i];
}
From last time

```
int m = 0;
co [i = 0 to n-1] {
  if (a[i] > m)
    m = a[i];
}
```

```
int m = 0;
co [i = 0 to n-1] {
  if (a[i] > m)
    if (a[i] > m)
      m = a[i];
}
```
Flow dependence (true dependence), Read-after-write (RAW): when an instruction depends on the results of a previous instruction.

Anti dependence, Write-after-read (WAR): an instruction requires a value that is later updated.

S1: \( x = 3 + i \)
S2: \( y = x \)
S3: \( x = 3 + i \)
S4: \( y = x \)
S5: \( z = y \)

S2 is dependent on S1.
S4 is dependent on S3, and S5 is dependent on S4 and also on S3.

S1: \( i = x \)
S2: \( x = 324 \)
S3: \( x = 42 \)
S4: \( y = x + 17 \)
S5: \( x = 6 \)

S2 is not dependent on S1, but re-ordering S2 and S1 would affect the final value of i.
S5 anti-depends on S4, because changing their ordering would affect the value of y.
From last time

**Flow dependence** (true dependence), Read-after-write (RAW): when an instruction depends on the results of a previous instruction.

![Flow dependence example](image)

**Anti dependence**, Write-after-read (WAR): an instruction requires a value that is later updated.

![Anti dependence example](image)

S2 is dependent on S1

S4 is dependent on S3, and S5 is dependent on S4 and also on S3

S2 is not dependent on S1, but re-ordering S2 and S1 would affect the final value of i

S5 anti-depends on S4, because changing their ordering would affect the value of y
Flow dependence (true dependence), Read-after-write (RAW): when an instruction depends on the results of a previous instruction.

Anti dependence, Write-after-read (WAR): an instruction requires a value that is later updated.

Q: How can we eliminate anti dependence from this code?
From last time

Flow dependence (true dependence), Read-after-write (RAW): when an instruction depends on the results of a previous instruction.

Anti dependence, Write-after-read (WAR): an instruction requires a value that is later updated.

Q: Does it now matter if S4 or S5 goes first?
Control Dependency Example
Code optimization
Warmup: Control Dependence

For which 2 ranges of $i$ sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

```plaintext
a = [ -2, -3, 3, 4, 5, 6 ]
for (i=1; i<n; i++){
    if (a[i-1] < 0){
        a[i] = -1;
    }
}
```
Warmup: Control Dependence

For which 2 ranges of i sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

In this scenario, what are the final values of the shared array a?

```
a = [-2, -3, 3, 4, 5, 6]
for (i=1; i<n; i++){
    if (a[i-1] < 0){
        a[i] = -1;
    }
}
```

```
for (i=1; i<3; i++){
    if (a[i-1] < 0){
        a[i] = -1;
    }
}
```

```
for (i=3; i<n; i++){
    if (a[i-1] < 0){
        a[i] = -1;
    }
}
```

```
for (i=3; i<n; i++){
    if (a[i-1] < 0){
        a[i] = -1;
    }
}
```
For which 2 ranges of \( i \) sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

In this scenario, what are the final values of the shared array \( a \)?

\[
a = [-2, -3, 3, 4, 5, 6] \]

\[
\begin{align*}
\text{Thread 1} \\
&\text{for } (i=1; i<n; i++)\
&\quad \text{if } (a[i-1] < 0)\
&\quad \quad a[i] = -1;\
&\end{align*}
\]

\[
\begin{align*}
\text{Thread 2} \\
&\text{for } (i=3; i<n; i++)\
&\quad \text{if } (a[i-1] < 0)\
&\quad \quad a[i] = -1;\
&\end{align*}
\]
For which 2 ranges of $i$ sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

$$a = [-2, -3, 3, 4, 5, 6]$$

For $i = 1$ to $n$:
- If $a[i-1] < 0$:
  - $a[i] = -1$;

If Thread 1 < Thread 2

If Thread 2 < Thread 1
Warmup : Control Dependence

For which 2 ranges of $i$ sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

\[
a = [-2, -3, 3, 4, 5, 6]
\]

for \(i=1; i<n; i++\)\
\[
\text{if } (a[i-1] < 0)\
\quad a[i] = -1;
\]
\}

For which 2 ranges of $i$ sent to two threads does there exist a history that exhibits control dependence when the threads are executed concurrently?

\[
a = [-2, -3, 3, 4, 5, 6]
\]

for \(i=1; i<n; i++\)\
\[
\text{if } (a[i-1] < 0)\
\quad a[i] = -1;
\]
\}

If Thread 1 < Thread 2

\[
a = [-2, -1, -1, -1, -1, -1, -1]
\]

If Thread 2 < Thread 1

\[
a = [-2, -1, -1, 4, 5, 6]
\]
Here, < means “completes”, thus Thread 2 < Thread 1 means that Thread 2 finishes before Thread 1 starts. If Thread 2 starts before Thread 1 starts, it may be possible in that scenario to get the same results as when Thread 1 finishes before Thread 2 starts.

If Thread 1 < Thread 2

\[
a = [-2, -3, 3, 4, 5, 6]
\]

If Thread 2 < Thread 1

\[
a = [-2, -1, -1, -1, 4, 5, 6]
\]
Q: Do Threads 1 and 2 cause race conditions on the entirety of array \( a \)?

\[
a = [-2, -3, 3, 4, 5, 6]
\]

for \( i=1; i<n; i++ \) {
  if \( a[i-1] < 0 \) {
    a[i] = -1;
  }
}

\[
i = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5
\]

\[
a = [-2, -3, \underline{3}, \underline{4}, 5, 6]
\]

Thread 1

for \( i=1; i<3; i++ \) {
  if \( a[i-1] < 0 \) {
    a[i] = -1;
  }
}

Thread 2

for \( i=3; i<n; i++ \) {
  if \( a[i-1] < 0 \) {
    a[i] = -1;
  }
}
Warmup: Control Dependence

Q: Do Threads 1 and 2 cause race conditions on the entirety of array \( a \)?

\[ a = [ -2, -3, 3, 4, 5, 6 ] \]

\[
\text{for } (i=1; i<n; i++)\{
    \text{if } (a[i-1] < 0)\{
        a[i] = -1;
    \}
\}
\]

Even though Threads 1 and 2 receive distinct index values of \( i \) of \( a \), race conditions exist nonetheless due to writing to an entry concurrently.

Q: What is that entry?
Q: If the output of a concurrent program is dependent on the history that is realized, how can you optimize the code for concurrent execution without using semaphores?
Code Optimization

Q: How can this code be optimized?

```
S0: y = 78;
S1: x = y + 32 - 16;
S2: y = 32 + 45.1 - 77;
S3: x = 45 * 12.7;
```
Q: How can this code be optimized?

Q: If our perspective is limited to thinking about sequential execution, can we eliminate this statement?

Q: If our perspective includes us reasoning about concurrent execution, can we eliminate this statement?
Code Optimization

Eliminate the dead code

This approach is suitable if the program is executed non-concurrently

But if the code is executed concurrently, what can go wrong?
Code Optimization

This approach is suitable if the program is executed non-concurrently.

But if the code is executed concurrently, what can go wrong?

Because of the histories that are possible, S2 may be executed BEFORE S1 and/or S0, which would alter the outcome of the variables’ values upon completion of all instructions.
When writing programs intended to be multi-thread executed in a concurrent environment, it is often a good idea to turn off compiler optimization ... hence optimization is the developer's responsibility.
Put yourself in the mindset of a compiler ... Q: how might this code be optimized?
Put yourself in the mindset of a compiler ... Q: how might this code be optimized?

Notice the redundancy ...
Put yourself in the mindset of a compiler … Q: how might this code be optimized?

Remove the redundancy by using the “output” of a previous calculation

Q: Has this optimization resulted in a program this is more or less multi-thread friendly when compared to the original version?
Put yourself in the mindset of a compiler ... Q: how might this code be optimized?

Remove the redundancy by using the "output" of a previous calculation.

Removing the redundant code has increased the number of dependencies, because now S3 is dependent on S1 AND S0, while in the previous case (code on the left), S3 was dependent on ONLY S0.

The use of waits, semaphores, etc. to impose these constraints might consume resources that will negate any improvements that you might expect by using threads.
Code Optimization

So if we cannot trust the compiler to optimize our code, how do WE do it?
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process.
Code Optimization

So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process

```c
int j = 2;
int k = 3;
for (int i = 0; i < N; i += 2) {
    a[i] = i * j + k;
}
```

Q: In this example, how can the code be optimized?
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process.

```c
int j = 2;
int k = 3;
for (int i = 0; i<N; i+=2){
    a[i] = i * j + k;
}
```

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop.
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop

```
int j = 2;
int k = 3;
for (int i = 0; i < N; i += 2) {
    a[i] = i * j + k;
}
```

Q: How many operations and loads are performed here?
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop

Q: How many operations and loads are performed here?

Two operations and 4 loads

```c
int j = 2;
int k = 3;
for (int i = 0; i < N; i += 2) {
    a[i] = i * j + k;
}
```
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop:

Two operations and 4 loads

One assignment

For a total of 7 “events”

Q: Can we reduce the number of events (assuming each event takes an equal number of clock cycles)
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process.

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop.

Yes ... notice that portions of the for loop’s body are “constant”.

How might the code be rewritten?
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process.

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop.

We’ve made the code “more complex,” but is it more efficient?
So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process.

Q: In this example, how can the code be optimized?

Think about this in terms of how many loads and operations are needed in the inner-most for loop.

Task: Count the number of "events"

1 operation
3 loads
2 assignments
Code Optimization

So if we cannot trust the compiler to optimize our code, how do WE do it?

Code optimization by a human is a time-consuming and creative process

```c
int j = 2;
int k = 3;
for (int i = 0; i < N; i += 2) {
    a[i] = i * j + k;
}
```

7 events

```c
int j = 2;
int k = 3;
int m = k;
int n = 2 * j;
for (int i = 0; i < N; i += 2) {
    a[i] = m;
    m = m + n;
}
```

6 events

Code restructuring is often counter-intuitive ... the revised for loop is visually MORE complex and larger, but it is more run-time optimal
```
for (int i=0; i<n; i++){
    if (a > 0){
        x(i) = x(i) + 1;
    }else{
        x(i) = 0;
    }
}
```

$x$ is an array, and $x(i)$ specifies $i^{th}$ index, and $a$ is some variable declared elsewhere.
Worksheet exercise 1

Q: How would you modify this for-loop, so that it does less “work”

```java
for (int i=0; i<n; i++){
    if (a > 0){
        x(i) = x(i) + 1;
    }else{
        x(i) = 0;
    }
}
```

Q: How many times is the conditional checked?
Code Optimization

Worksheet exercise 1

Q: How would you modify this for-loop, so that it does less "work"

```plaintext
for (int i=0; i<n; i++){
    if (a > 0){
        x(i) = x(i) + 1;
    } else{
        x(i) = 0;
    }
}
```

Q: How many times is the conditional checked?

Test promotion (loop unswitching) : move a loop-independent test OUT of the loop
Code Optimization

Worksheet exercise 1

Q: How would you modify this for-loop, so that it does less “work”

```java
for (int i=0; i<n; i++){
    if (a > 0){
        x(i) = x(i) + 1;
    }else{
        x(i) = 0;
    }
}
```

Q: How many times is the conditional checked?

Test promotion (loop unswitching) : move a loop-independent test OUT of the loop

Loop independent test is moved OUT of the for loop

```java
if (a > 0){
    for (int i=0; i<n; i++){
        x(i) = x(i) + 1;
    }
}else{
    for (int i=0; i<n; i++){
        x(i) = 0;
    }
}
```

Test promotion allows for easier loop parallelization
Worksheet exercise 2

Q: How would you modify this for-loop, so that it does less “work”

```c
int n = 500000;
for (int i=0; i<n; i++){
    if (i == 0){
        x(i) = 0;
    }else if (i == n){
        x(i) = n;
    }else{
        x(i) = x(i) + y(i);
    }
}
```
Worksheet exercise 2

Q: How would you modify this for-loop, so that it does less “work”

```c
int n = 500000;
for (int i=0; i<n; i++){
    if (i == 0){
        x(i) = 0;
    }else if (i == n){
        x(i) = n;
    }else{
        x(i) = x(i) + y(i);
    }
}
```

If n is large, how often are the different parts of the for-loop's body evaluated?
Code Optimization

Worksheet exercise 2

Q: How would you modify this for-loop, so that it does less “work”

```c
int n = 500000;
for (int i=0; i<n; i++){
    if (i == 0){
        x(i) = 0;
    }else if (i == n){
        x(i) = n;
    }else{
        x(i) = x(i) + y(i);
    }
}
```

If n is large, how often are the different parts of the for-loop's body evaluated?

Loop peeling: If numerical code deals with boundary conditions in the first or last iterations of a loop, then those components can be taken out of the loop.
Loop peeling: If numerical code deals with boundary conditions in the first or last iterations of a loop, then those components can be taken out of the loop.

These if and else evaluate to True at the boundary conditions.
Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```
Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```

Notice that the for-loop uses three “items”: i, a[i], and b[i]
Code Optimization

Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```

Notice that the for-loop uses three “items”: i, a[i], and b[i]

Locality of reference is poor.

For each iteration of the for loop a cache miss occurs, and b[i] must be evicted to make room for a[i]. It is retrieved (swapped in) almost immediately at the next iteration of the for loop.
Code Optimization

Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```

Notice that the for-loop uses three “items”: i, a[i], and b[i]

Loop fission: a loop is broken into multiple “smaller” loops over the same index range, and each “smaller” loop performs only a part of the original loop’s body.
Loop fission: a loop is broken into multiple “smaller” loops over the same index range, and each “smaller” loop performs only a part of the original loop’s body.

Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```

Notice that the for-loop uses three “items”: i, a[i], and b[i].

Two smaller loops, each of which has a MUCH better locality of reference:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
}

for (i=0; i<600; i++){
    b[i] = 47;
}
```
Loop fission: a loop is broken into multiple “smaller” loops over the same index range, and each “smaller” loop performs only a part of the original loop’s body.

Notice that the “optimized” code is lengthier, but it potentially runs MUCH faster if cache misses are minimized.

Worksheet exercise 3

Assuming the cache can hold only two “items”, how would you improve the below code:

```c
int i;
int a[600];
int b[600];
for (i=0; i<600; i++){
    a[i] = 45;
    b[i] = 47;
}
```

Notice that the for-loop uses three “items”: i, a[i], and b[i]
Q: How does a compiler perform loop optimization?

Q: What is the “algorithm”? 
Q: How does a compiler perform loop optimization?

Q: What is the “algorithm”?

Q: How would you optimize this for loop?

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Worksheet exercise 4
Q: How does a compiler perform loop optimization?

Q: What is the “algorithm”?

Q: How would you optimize this for loop?

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

You could unroll the entire loop, and have 600 sequential executions of the 4 lines of code that appear in the body of the for loop ... but would that maintain dependencies?
Step 1: Determine dependencies

Task: Enumerate all of the dependencies
Code Optimization

Step 1: Determine dependencies

- S1 is dependent on S2

But it is not that simple, because S2 is dependent on S3 ... etc.

```
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Step 1: Determine dependencies
- S1 is dependent on S2
- S2 is dependent on ...
- S3 is dependent on ...
- S4 is dependent on ...
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Step 1: Determine dependencies
- S1 is dependent on S2
- S2 is dependent on S3
- S3 is dependent on S2
- S4 is dependent on S3
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++)
{
    a[i] = a[i] + b[i];
    b[i] = c[i]*x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Step 1: Determine dependencies
- S1 is dependent on S2
- S2 is dependent on S3
- S3 is dependent on S2
- S4 is dependent on S3

Step 2: Build (possibly cyclic) graph
Q: How can the graph be used to infer which statement should go first, second, last, etc.?
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Step 1: Determine dependencies
- S1 is dependent on S2
- S2 is dependent on S3
- S3 is dependent on S2
- S4 is dependent on S3

Step 2: Build (possibly cyclic) graph

Step 3: condense cycles
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++) {
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Step 1: Determine dependencies
- S1 is dependent on S2
- S2 is dependent on S3
- S3 is dependent on S2
- S4 is dependent on S3

Step 2: Build (possibly cyclic) graph

Step 3: condense cycles

Q: Now what?
Q: How do we use the final graph to determine the order of executions to inform us how the original problem can be broken into smaller ones?
Code Optimization

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
    d[i] = Math.sqrt(c[i]);
}
```

Each node corresponds to a fission
The “solution” may not be unique (here the order of execution for S4 or S1 can be interchanged), and you must be careful!

Q: Is the above the correct ordering of loops?
Q: Should the yellow loop proceed before or after the green loop?
Code Optimization

At the \(i\)th iteration of the for loop, S1 reads from \(b[i]\), and then immediately \(b[i]\) is altered (S2).

Q: Does this help us impose a specific order of execution of the fused loops?
Code Optimization

If the green loop were to complete on a thread before the yellow loop begins, then the original code (left) might produce an array $a$ that is different than the code on the right.

Q: How should the fused loops be executed?
If the green loop were to complete on a thread before the yellow loop begins, then the original code (left) might produce an array `a` that is different than the code on the right.

Q: How should the fused loops be executed?

The yellow loop should proceed first, then the green loop, then the blue loop.

Q: Must the entirety of the yellow loop complete prior to the green loop starting?
In this single for loop approach ...

```
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
}
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
}
for (i=0; i<600; i++){
    d[i] = Math.sqrt(c[i]);
}
```
In this single for loop approach ...

First $a[0]$ is calculated
In this single for loop approach ...

Then b[0]
Code Optimization

In this single for loop approach ...

Then c[0]
In this single for loop approach ...

Then d[0]
In this single for loop approach ...

Then $a[1]$ ... etc. etc.

```
int a[], b[], c[], d[];
for (i=0; i<600; i++)
{
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
}
for (i=0; i<600; i++)
{
    a[i] = a[i] + b[i];
}
for (i=0; i<600; i++)
{
    d[i] = Math.sqrt(c[i]);
}
```
In this single for loop approach ...

Then a[1] ... etc. etc.

In the fused loop approach, if you have a computer with n > 3 processors ...

Q: Which loop do you send to which thread, and how do you schedule them?
Q: If we schedule the processing of the arrays as shown below, does that guarantee that the yellow “loop” finishes before the green? And the green before the blue?

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
}
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
}
for (i=0; i<600; i++){
    d[i] = Math.sqrt(c[i]);
}
```
Q: If we schedule the processing of the arrays as shown below, does that guarantee that the yellow “loop” finishes before the green? And the green before the blue?

Q: Can we do better?
Q: If we delay the start of the CPU that is handling the green loop so that it begins after the CPU that is handling the yellow loop begins, and we delay the start of the CPU that is handling the blue loop until a bit after the CPU that is handling the green loop begins, are the calculations “safe”?

```java
int a[], b[], c[], d[];
for (i=0; i<600; i++){
    b[i] = c[i] * x + y;
    c[i] = 1/b[i];
}
for (i=0; i<600; i++){
    a[i] = a[i] + b[i];
}
for (i=0; i<600; i++){
    d[i] = Math.sqrt(c[i]);
}
```
Q: If we delay the start of the CPU that is handling the green loop so that it begins after the CPU that is handling the yellow loop begins, and we delay the start of the CPU that is handling the blue loop until a bit after the CPU that is handling the green loop begins, are the calculations “safe”?

No. Just because the CPU that processes the yellow loop begins BEFORE the CPU that is processing the green loop, that does not guarantee that the “green” CPU will not overtake the “yellow” CPU.

Q: What is the solution?
Up Next

Locks
Critical Sections
Barriers