INFORMATION and REMINDERS

• Homework 7 is due TODAY, December 1 at 11:59pm.

• New second programming project deadline: Monday, December 4, 11:59pm.

• Midterm sample solutions are available. 
  Midterm grade challenge deadline: 
  December 11.

• Final exam: Wednesday, December 20, 4:00-7:00pm, in our classroom.

DO YOU HAVE A CONFLICT?

http://sasundergrad.rutgers.edu/forms/final-exam-conflict

  – More than two (2) final exams on one calendar day
  – More than two (2) final exams scheduled in consecutive periods
  – Two final exams scheduled for the same exam period.
Dependence — Overview

Definition — There is a data dependence from statement \( S_1 \) to statement \( S_2 \) \((S_1 \delta S_2)\) if

1. Both statements access the same memory location, and
2. There is a run-time execution path from \( S_1 \) to \( S_2 \).

Data dependence classification

“\( S_2 \) depends on \( S_1 \)” — \( S_1 \delta S_2 \)

true (flow) dependence
occurs when \( S_1 \) writes a memory location that \( S_2 \) later reads

anti dependence
occurs when \( S_1 \) reads a memory location that \( S_2 \) later writes

output dependence
occurs when \( S_1 \) writes a memory location that \( S_2 \) later writes

input dependence
occurs when \( S_1 \) reads a memory location that \( S_2 \) later reads. Note: Input dependences do not restrict statement \((load/store)\) order!
Dependence — Where do we need it?

We restrict our discussion to data dependence for scalar and subscripted variables (no pointers and no control dependence).

Examples:

\begin{verbatim}
do I = 1, 100 do I = 1, 99
do J = 1, 100 do J = 1, 100
   A(I,J) = A(I,J) + 1 A(I,J) = A(I+1,J) + 1
 enddo   enddo
enddo   enddo
\end{verbatim}

**vectorization**

\begin{verbatim}
A(1:100:1,1:100:1) = A(1:100:1,1:100:1) + 1
A(1:99,1:100) = A(2:100,1:100) + 1
\end{verbatim}

**parallelization**

\begin{verbatim}
doall I = 1, 100 do I = 1, 99
   doall J = 1, 100 doall J = 1, 100
      A(I,J) = A(I,J) + 1 A(I,J) = A(I+1,J) + 1
     enddo   enddo
      \textit{implicit barrier sync.} \textit{implicit barrier sync.}
     enddo   enddo
      \textit{implicit barrier sync.} \textit{implicit barrier sync.}
\end{verbatim}
Vectorization vs. Parallelization

**vectorization** — Find parallelism in innermost loops; fine-grain parallelism

**parallelization** — Find parallelism in outermost loops; coarse-grain parallelism

- Parallelization is considered more complex than vectorization, since finding coarse-grain parallelism requires more analysis (e.g., interprocedural analysis).

- Automatic vectorizers have been very successful
A **loop-independent** dependence exists regardless of the loop structure. The source and sink of the dependence occur on the same loop iteration.

A **loop-carried** dependence is induced by the iterations of a loop. The source and sink of the dependence occur on different loop iterations.

*Loop-carried dependences can inhibit parallelization and loop transformations*
Dependence Testing

Given

\[
\begin{align*}
\text{do } & i_1 = L_1, U_1 \\
\ldots \\
\text{do } & i_n = L_n, U_n \\
S_1 & \quad A(f_1(i_1, \ldots, i_n), \ldots, f_m(i_1, \ldots, i_n)) = \ldots \\
S_2 & \quad \ldots = A(g_1(i_1, \ldots, i_n), \ldots, g_m(i_1, \ldots, i_n))
\end{align*}
\]

A dependence between statement $S_1$ and $S_2$, denoted $S_1 \delta S_2$, indicates that $S_1$, the source, must be executed before $S_2$, the sink on some iteration of the nest.

Let $\alpha$ & $\beta$ be a vector of $n$ integers within the ranges of the lower and upper bounds of the $n$ loops.

Does $\exists \alpha \leq \beta$, s.t.

\[
f_k(\alpha) = g_k(\beta) \quad \forall k, \ 1 \leq k \leq m \ ?
\]
Iteration Space

\[
\begin{align*}
\text{do } I &= 1, 5 \\
\text{do } J &= I, 6 \\
\quad \ldots \\
\text{enddo} \\
\text{enddo}
\end{align*}
\]

\[
1 \leq I \leq 5 \\
I \leq J \leq 6
\]

- lexicographical (sequential) order for the above iteration space is

\[
(1,1), (1,2), \ldots, (1,6) \\
(2,2), (2,3), \ldots (2,6) \\
\ldots \n(5,5), (5,6)
\]

- given \( I = (i_1, \ldots i_n) \) and \( I' = (i'_1, \ldots, i'_n) \),

\[
I < I' \text{ iff } (i_1, i_2, \ldots i_k) = (i'_1, i'_2, \ldots i'_k) \land i_{k+1} < i'_{k+1}
\]
Distance & Direction Vectors

Distance Vector = number of iterations between accesses to the same location
Direction Vector = direction in iteration space (|=,<,>)

\[
S_1 \delta S_1 \\
S_2 \delta S_2 \\
S_3 \delta S_3
\]
Next Lecture

Things to do:

- Dependence testing
- More on automatic vectorization / parallelization