Procedural dependency

- Can not execute instructions after a branch until the branch is executed. (effect of a branch on a superscalar pipeline of degree 2, I1 is a branch)

\[
\begin{align*}
I0: & F D E W \\
I1: & F D E W \\
I2: & F D E W \\
I3: & F D E W \\
I4: & F D E W \\
I5: & F D E W \\
\end{align*}
\]

Resource conflict

- Two or more instructions requiring access to the same resource at the same time. (e.g. two arithmetic instructions use ALU adder)

\[
\begin{align*}
I0: & F D E W \\
I1: & F D E W \\
\end{align*}
\]

(similar to true data dependency, but true data dependency is unavoidable)

- Can be eliminated by duplication of resources (e.g. have two arithmetic units)

Output dependency (WAW)

\[
\begin{align*}
I0: & R3 \leftarrow R3 \text{ op } R5 \\
I1: & R4 \leftarrow R3 + 1 \\
I2: & R3 \leftarrow R5 + 1 \\
I3: & R7 \leftarrow R3 \text{ op } R4 \\
\end{align*}
\]

- True data dependency: I1 depends on the result of I0, I3 depends on the result of I2
- Output dependency: if I2 completes before I0, the contents of R3 will be wrong to I3.

Anti-dependency (Must R before W, WAR)

\[
\begin{align*}
I0: & R3 \leftarrow R3 \text{ op } R5 \\
I1: & R4 \leftarrow R3 + 1 \\
I2: & R3 \leftarrow R5 + 1 \\
I3: & R7 \leftarrow R3 \text{ op } R4 \\
\end{align*}
\]

- I2 can NOT complete before I1 starts, since I1 needs a value in R3 and I2 changes R3.

Register Renaming

- A method to address output dependency and anti-dependency.
Output and anti-dependency are examples of storage conflicts. Duplication of resources can address it.

In essence, registers are allocated dynamically by the processor hardware. (registers are not specifically named)
- When a new register value is created, a new register is allocated for that value.
- Subsequent instructions that access that value must be revised to refer to the register containing the value. (Renaming)
- The same original register reference in several different instructions may refer to different actual registers, if different values are intended.

Example:

\[
\begin{align*}
I_0: & \quad R3 \leftarrow R3 \text{ op } R5 \\
I_1: & \quad R4 \leftarrow R3 + 1 \\
I_2: & \quad R3 \leftarrow R5 + 1 \\
I_3: & \quad R7 \leftarrow R3 \text{ op } R4 \\
\end{align*}
\]

\[
\begin{align*}
I_0: & \quad R3_a \leftarrow R3 \text{ op } R5 \\
I_1: & \quad R4_a \leftarrow R3_a + 1 \\
I_2: & \quad R3_b \leftarrow R5 + 1 \\
I_3: & \quad R7_a \leftarrow R3_b \text{ op } R4_a \\
\end{align*}
\]

- The register reference with the subscript refers to a hardware register allocated to hold a new value