Two processes sharing data:

\[
\begin{align*}
X & \text{ DC.L 7} \\
Y_1 & \text{ DC.L 5} \\
Y_2 & \text{ DC.L 3}
\end{align*}
\]

\[
\begin{align*}
p1: & \\
move.l & X, D0 \quad ; s1 \\
add.l & Y_1, D0 \quad ; s2 \\
move.l & D0, X \quad ; s3
\end{align*}
\]

\[
\begin{align*}
p2: & \\
move.l & X, D0 \quad ; t1 \\
add.l & Y_2, D0 \quad ; t2 \\
move.l & D0, X \quad ; t3
\end{align*}
\]

Order of execution:

- \( s1, s2, s3, t1, t2, t3 \ - X = 15 \)
- \( t1, t2, t3, s1, s2, s3 \ - X = 15 \)
- \( s1, t1, t2, t3, s2, s3 \ - X = 12 \)
- \( t1, s1, s2, s3, t2, t3 \ - X = 10 \)

Using a single flag.

Flag = 1 indicates the resource is busy, Flag = 0 indicates the resource is free. Initially Flag = 0. Consider the following sequences of code:

Process 1

\[
\begin{align*}
P1: & \\
\text{Check1:} & \text{ cmpi.b #1, Flag} \\
& \text{ beq Check1} \\
& \text{ move.b #1, Flag} \\
\ldots & \\
& \text{ Critical Section} \\
\ldots & \\
& \text{ clr.b Flag}
\end{align*}
\]

Process 2

\[
\begin{align*}
P2: & \\
\text{Check2:} & \text{ cmpi.b #1, Flag} \\
& \text{ beq Check2} \\
& \text{ mov.b #1, Flag} \\
\ldots & \\
& \text{ Critical Section} \\
\ldots & \\
& \text{ clr.b Flag}
\end{align*}
\]
Using 2 flags.

Flag1 and Flag2, one for each process. Again 1 means in the critical section, 0 means not.

The code would look like:

```
Using a test-and-set instruction.
```

In the 68000, the **TAS** (Test and Set) instruction is used. This instruction tests a memory location and sets condition codes, it then sets bit 7 to a 1.

```
Using a test-and-set instruction.
```

In the 68000, the **TAS** (Test and Set) instruction is used. This instruction tests a memory location and sets condition codes, it then sets bit 7 to a 1.