

Two processes sharing data:

```
X DC.L 7
Y1 DC.L 5
Y2 DC.L 3
```

```
p1:      move.l  X, D0    ; s1
         add.l   Y1, D0   ; s2
         move.l  D0, X    ; s3

p2:      move.l  X, D0    ; t1
         add.l   Y2, D0   ; t2
         move.l  D0, X    ; t3
```

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
P1:      ...
Check1:  cmpi.b  #1,Flag
         beq     Check1
         move.b  #1, Flag
         ...
         Critical Section
         ...
         clr.b   Flag
```

```
Process 2
P2:      ...
Check2:  cmpi.b  #1,Flag
         beq     Check2
         mov.b   #1,Flag
         ...
         Critical Section
         ...
         clr.b   Flag
```

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:

```
Process 1
P1:  ...
      move.b  #1,Flag1
Check1:  cmpi.b #1,Flag2
          beq   Check1
          ...
          Critical section
          ...
          clr.b  Flag1

Process 2
P2:  ...
      move.b  #1,Flag2
Check2:  cmpi.b #1,Flag1
          beq   Check2
          ...
          Critical section
          ...
          clr.b  Flag2
```

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.

```
Process n
Pn:  ...
Check1:  tas   Flag
          bne  Check1
          ...
          Critical Section
          ...
          clr.b  Flag
```