

```

// *****
// SOLUZIONE 1
// - Disattivazione degli interrupt
// *****
main()
{
    // ...
    << disattivazione interrupt (asm CLI) >>
    << sezione critica >>
    << riattivazione interrupt (asm STI) >>
    // ...
}

// *****
// SOLUZIONE 2
// - Variabile libero
// *****
int libero = 1;

// Processo 1
main()
{
    // ...
    while (!libero);
    libero = 0;
    << sezione critica >>
    libero = 1;
    // ...
}

// Processo 2
main()
{
    // ...
    while (!libero);
    libero = 0;
    << sezione critica >>
    libero = 1;
    // ...
}

// *****
// SOLUZIONE 3
// - Variabile turno
// *****
int turno = 1; // turno IN (1, 2)

// Processo 1
main()
{
    // ...
    while (turno != 1);
    << sezione critica >>;
    turno = 2;
    // ...
}

```

```

// Processo 2
main()
{
    // ...
    while (turno != 2);
    << sezione critica >>;
    turno = 1;
    // ...
}

// *****
// SOLUZIONE 4
// - 2 variabili libero
// *****
int libero1 = 0;
int libero2 = 0;

// Processo 1
main()
{
    // ...
    libero1 = 1;
    while (!libero2);
    << sezione critica >>
    libero1 = 0;
    // ...
}

// Processo 2
main()
{
    // ...
    libero2 = 1;
    while (!libero1);
    << sezione critica >>
    libero2 = 0;
    // ...
}

// *****
// SOLUZIONE 5
// - 2 variabili libero con controllo reciproco
// *****
int libero1 = 0;
int libero2 = 0;

// Processo 1
main()
{
    // ...
    libero1 = 1;
    while (libero2)
    {
        libero1 = 0;
        while (libero2);
        libero1 = 1;
    }
    << sezione critica >>

```

```

libero1 = 0;
// ...
}

// Processo 2
main()
{
// ...
libero2 = 1;
while (libero1)
{
libero2 = 0;
while (libero1);
libero2 = 1;
}
<< sezione critica >>
libero2 = 0;
// ...
}
// *****
// SOLUZIONE 6
// - Algoritmo di Dekker
// *****
int libero1 = 0;
int libero2 = 0;
int turno = 1; // turno IN (1, 2)

// Processo 1
main()
{
// ...
libero1 = 1;
while (libero2)
{
if (turno == 2)
{
libero1 = 0;
while (turno != 1);
libero1 = 1;
}
}
<< sezione critica >>
turno = 2;
libero1 = 0;
// ...
}

// Processo 2
main()
{
// ...
libero2 = 1;
while (libero1)
{
if (turno == 1)
{
libero2 = 0;
while (turno != 2);
}
}
}

```

```

        libero2 = 1;
    }
}
<< sezione critica >>
turno = 1;
libero2 = 0;
// ...
}

// *****
// SOLUZIONE 7
// - Algoritmo di Peterson
// *****
int libero1 = 0;
int libero2 = 0;
int turno = 1; // turno IN (1, 2)

// Processo 1
main()
{
    // ...
    libero1 = 1;
    turno = 2;
    while (libero2 && turno == 2);
    << sezione critica >>
    libero1 = 0;
    // ...
}

// Processo 2
main()
{
    // ...
    libero2 = 1;
    turno = 1;
    while (libero1 && turno == 1);
    << sezione critica >>
    libero2 = 0;
    // ...
}

// *****
// SOLUZIONE HARDWARE
// - Lock
// *****
int x = 1;

// Processo 1
main()
{
    // ...
    lock (&x);
    << sezione critica >>;
    unlock (&x);
    // ...
}

```

```

// Processo 2
main()
{
    // ...
    lock (&x);
    << sezione critica >>;
    unlock (&x);
    // ...
}

// *****
// SOLUZIONE HARDWARE
// - Comportamento logico di lock e unlock
// *****
void lock(int* x)
{
    while(!*x);
    *x = 0;
}

void unlock(int* x)
{
    *x = 1;
}

// Realizzazione lock con Test&Set
void lock (int* x)
{
    while(!TestAndSet(x));
}

// Realizzazione TestAndSet: realizzata dall'ISA in un solo ciclo di memoria!
int TestAndSet(int* a)
{
    int R = *a;
    *a = 0;
    return R;
}

// Realizzazione TestAndSet ASM: realizzata dall'ISA in un solo ciclo di memoria!
lock(x):
    tsl register, x    // register <- x, x <-- 0
    cmp register, 1    // register == 1?
    jne lock           // se x = 0 ricomincio
    ret                // termino

// Realizzazione lock con Exchange
void lock (int* x)
{
    int priv = 0;
    do
    {
        EXCH(x, &priv);
    } while (priv == 0);
}

// Realizzazione Exchange: realizzata dall'ISA in un solo ciclo di memoria!
int EXCH(int* a, int* b)
{

```

```
int temp = *a;  
*a = *b;  
*b = temp;  
}
```