

1. naloga: Take the multiprocessor with 30 processors, each processor is capable of max 2Gflops. What is the maximum capability of the multiprocessor for the execution of an application that has 5% sequential code?

$$\begin{aligned} \frac{1}{B+\frac{1-B}{P}} &= S(n) = \frac{T}{\frac{T(1-s)}{n} + Ts} \quad S = 5\% = \frac{5}{100} \\ S(30) &= \frac{T}{\frac{T(\frac{95}{100})}{30} + T \cdot \frac{5}{100}} = \frac{1}{\frac{95}{30 \cdot 100} + \frac{5 \cdot 30}{100 \cdot 30}} \\ \frac{1}{\frac{245}{3000}} &= \frac{3000}{245} = 12.24 \text{ times faster} \\ \text{So } 2, 12, 24 &= 24.48 \text{ floating point operations per second} \end{aligned}$$

2. naloga: Give the definition of one-time property, for each thread prove if this property holds:

```
int x = 0, y = 10, z = 6
co x = x + z;
|| y = x - 1;
|| z = z + 16;
oc
```

At most once property

contains 1 cr and read by other. ✓

contains 1 cr and y read by other. ✓

contains 0 cr and z read by other. ✓

cr = critical reference = reference to a variable that is changed by other process.

3. naloga:

```
int x = 0; y = 1; z = 2;
co z = x + 3 + y;
|| x = z + y;
oc
```

x	y	z	
1	1	2	{5, 1, 4}
1	2	1	{3, 1, 4}
1	2	2	{3, 1, 4}
2	1	1	{3, 1, 4}
2	1	2	{3, 1, 4}
2	2	1	{3, 1, 7}

List all possible program paths and for each group of paths list the final values.

4. naloga: Look at the routine:

```
co write("1"); write("2");
|| write("3"); write("4");
|| write("5"); write("6");
oc
```

1. How many different prints can this program produce? Why!

If there are n threads such that $Thread_i$ executes m_i atomic actions, then the number of possible interleavings (options/solutions) of the atomic actions is:

$$\frac{(m_1 + m_2 + \dots + m_n)!}{m_1! \cdot m_2! \cdot \dots \cdot m_n!}$$

$$\begin{aligned} \frac{(2+2+2)!}{2! \cdot 2! \cdot 2!} &= \frac{6!}{2! \cdot 2! \cdot 2!} \\ &= 90 \end{aligned}$$

2. Use semaphores in such a way that the program always prints: 1,2,3,4,5,6.

```
int sem s1=0, s2=0;
co write("1"); write("2"); v(s1);
|| p(s1); write("3"); write("4"); v(s2);
|| p(s2); write("5"); write("6"); // v increases semaphore, p decreases.
oc
```