Laboratorium 3. Cel: Zastosowanie szeregowania afinicznego do znalezienia równoległości pozbawionej synchronizacji (patrz wykład 9), 2 godziny.

Zadania:

1. Dla wskazanej pętli za pomocą kalkulatora ISCC znaleźć relację zależności, R, przestrzeń iteracji, LD, oraz zrobić rysunek grafu zależności w przestrzeni 6 x 6. W tym celu trzeba zastosować operator scan (R*[n]->{:n=6}); który wygeneruje wszystkie zależności w przestrzeni 6 x 6, pierwsza krotka wskazuje początek zależności (strzałki), druga krotka – koniec zależności (strzałki).
   !!!Uwaga: dla niektórych pętli w przestrzeni 6x6 zależności mogą nie istnieć, w takim przypadku należy rozszerzyć przestrzeń do rozmiaru 12x12.
2. Za pomocą operatora kalkulatora ISCC: *IslSchedule := schedule LD respecting R minimizing R* znaleźć szeregowanie afiniczne w postaci drzewa (patrz wykład 4).
3. Za pomocą operatora kalkulatora ISCC: *map* przekonwertować szeregowanie afiniczne w postaci drzewa na szeregowanie w postaci relacji.
4. Sprawdzić które (tylko jedno) z uzyskanych szeregowań pozwala na ekstrakcję równoległości pozbawionej synchronizacji. Stosując właściwe szeregowanie wygenerować pseudokod i kod kompilowalny reprezentujący równoległość pozbawioną synchronizacji.

   Celem sprawdzenia czy jest szeregowanie pozwalające na wygenerowania kodu bez synchronizacji trzeba wygenerować wektory dystansu stosując instrukcję iscc „deltas R",

   gdzie R jest to relacja zależności. Załóżmy, że szeregowania są jak niżej

   (c11, c12), (c21,c21),

   natomiast wektory dystansu to są

   (d11, d12), (d21,d22) //uwaga: może być tylko jeden wektor dystansu!!!

   Żeby sprawdzić, czy jest równoległość pozbawiona synchronizacji, sprawdzamy najpierw pierwsze szeregowanie. Jeśli c11*d11+c12*d12=0 oraz c11*d21+c12*d22=0 to szeregowanie pozwala na wygenerowania kodu bez synchronizacji, wykorzystujemy go do wygenerowania kodu.

   Natomiast jeśli c11*d11+c12*d12 nie jest 0 lub c11*d21+c12*d22 nie jest 0, to wtedy analizujemy drugie szeregowania, czyli sprawdzamy:

   Czy c21*d11+c22*d12=0 oraz c21*d21+c22*d22=0

   Jeśli tak, to wybieramy drugie szeregowania do wygenerowania kodu bez synchronizacji, inaczej brak jest równoległości pozbawionej synchronizacji.

   W oparciu o wybrane szeregowanie utworzyć relację CODE_SYNCH_FREE (patrz L3).

5. Stosując uzyskaną relację CODE_SYNCH_FREE za pomocą operatora *scan* znaleźć wszystkie niezależne fragmenty kodu i zaznaczyć je na rysunku stworzonym w p. 1 (rysunek z zależnościami) w przestrzeni 6x6 (12x12 jeśli przestrzeń została rozszerzona, patrz p.1).

6. Wygenerować pseudokod i kod kompilowalny implementujący równoległość pozbawioną synchronizacji.

!!!!Uwaga W kodzie pozbawionym synchronizacji, pierwsza pętla jest równoległa

7. Zastosować program do porównania produkowanych przez pętle wyników (zadanie z poprzedniego laboratorium) do sprawdzenia poprawności kodu docelowego w przestrzeni 6x6.
8. Opracować sprawozdanie.

Patrz skrypt L3 pokazujący dla przykładowej pętli realizację poszczególnych zadań wyżej.

Warianty pętli:

1.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[i][j-1];
```
2.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i][j-2];
```
3.
```
for(i=1;i<=n;i++)
  for(j=3;j<=n;j++)
    a[i][j] = a[i][j-3];
```
4.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[i-1][j-1];
```
5.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j-1];
```
6.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j-2];
```
7.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j+2];
```
8.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i-1][j+2];
```
9.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
```

```
    a[i][j] = a[i-1][j+1];
```

10.

```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i+3][j+4];
```

11.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+3][j-4];
```

12.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+4][j-4];
```

13.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-4];
```

14.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[2i+5][j-4];
```

15.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[2i+5][j-3];
```

16.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[2i+4][j-3];
```

Sprawozdanie powinno zawierać: pętlę, skrypt implementujący zadania oraz wyniki wszystkich zadań łącznie z wynikami porównania danych wyjściowych produkowanych przez program oryginalny i program docelowy.