	Politechnika Warszawska	***Programowanie równoległe i rozproszone***
Data: ***12.01.2018***	***Wykonawca:*** Królikowski Krzysztof (244739)	***OpenMP - Zadanie JB5.*** ***Rozwiązywanie układów równań liniowych metodą Jacobiego***

1. Zrównoleglenie programu (OpenMP).

Zmiany względem części sekwencyjnej polegały na dodaniu odpowiednich komend zrownoleglających z biblioteki OpenMP do funkcji *jacobi*. Zrównolegleniu podlega wyznaczenie elementów wektora $\boldsymbol{x_k}$, wyznaczonego dla k-tej iteracji algorytmu. Innymi słowy macierz $\boldsymbol{A}$ jest podzielona na poziome pasy, a każdym z pasów zajmują się osobne wątki. Kluczowy listing funkcjonalności zaprezentowano poniżej.

```
do //do_while
{
   k = k + 1;
   #pragma omp parallel default(none) shared(A, cols, rows, x_0, x_, b, serr)
   {
      #pragma omp single
      {
         serr = 0; //[BUG FIXED][1]
      }
      #pragma omp for private(i, j, ax)
      for(i=0; i<rows; i++)
      {
         ax = 0;
         for(j=0; j<cols; j++)
            if(i!=j)
               ax = ax + (A[i][j]*x_0[j]);
         x_[i] = (-ax + b[i])/A[i][i];
      }
      #pragma omp for private(i) reduction(+:serr)
      for(i=0; i<rows; i++)
      {
         serr += pow(x_[i]-x_0[i],2); //[BUG FIXED][1]
         x_0[i]= x_[i];
      }
   }//omp parallel
   rserr = sqrt(serr);
} while(rserr>epsilon & k < max_it);
```

[1] Naprawiony błąd względem poprzedniej wersji programu

1.1 Składnia OpenMP

```
# pragma omp parallel <klauzule >
{
   <blok instrukcji >
}
```

Dyrektywa ta definiuje blok równoległy. Przekazuje kompilatorowi informację, że blok instrukcji ma być wykonywany przez wiele niezależnych wątków. Użyte w programie klauzule oznaczają odpowiednio:

- *default(none)* – deklaracja, że wszystkie zmienne widziane z danego bloku równoległego muszą być zadeklarowane w sposób jawny (shared/private),
- *shared(...)* – deklaracja zmiennych współdzielonych, czyli wspólnych dla wszystkich wątków

```
# pragma omp single <klauzule >
{
   <blok instrukcji >
}
```

Dyrektywa oznaczająca lokalną jednowatkowść, to znaczy dana część kodu jest wykonywana przez jeden watek. Pozostałe watki oczekują na skończenie pracy przez ten jeden wątek.

```
# pragma omp for <klauzule >
<pętla for >
```

Dyrektywa rozdzielająca pracę procesora między wątki. Mówi, że poszczególne iteracje pętli *for*, w bloku tej instrukcji, mają być wykonywane niezależnie od siebie przez wiele wątków. Użyte w programie klauzule oznaczają odpowiednio:

- *private(...)* – deklaracja zmiennych prywatnych dla kazdego wątku, co oznacza, że na potrzzeby każdego wątku tworzona jest zmienna lokalna do której pozostałe wątki nie maja dostępu,
- reduction(+:*zmienna*) – ten zapis oznacza, że podczas obliczeń tworzona jest lokalna kopia zmiennej *zmienna* na każdym wątku. Na koniec pętli wszystkie lokalne *zmienna* są dodawane

2. Wynik zrównoleglenia

Zrównoleglanie wykonano na procesorze Intel core i5 2410M (2 rdzenie fizyczne/4 wątki) z 8GB pamięci RAM. Program zmodyfikowano tak, aby można było wprowadzać do programu liczbę wątków użytych w zrównolegleniu: *omp_set_num_threads(numer_of_threads)* z wiersza poleceń.

Sposób kompilacji:

gcc -fopenmp -march=native -O2 "auxs.c" "jacobi.c" -o pjacobi.exe

Tabela 1. Porównanie czasów [s] wykonania programu w wersji sekwencyjnej oraz przy zastosowaniu OpenMP ze zmienną liczbą wykorzystanych wątków

Rozmiar problemu [liczba wierszy macierzy]	*Część sekwencyjna*	*OpenMP nth=1*	*OpenMP nth=2*	*OpenMP nth=3*	*OpenMp nth=4*
10000	116,346	116,048	76,518	60,003	50,502
5000	28,189	28,283	15,407	13,251	12,076
1000	1,093	1,138	0,656	0,609	0,531

Współczynnik przyspieszenia jest definiowany jako:

$$S(n,p) = \frac{T(n,1)}{T(n,p)}$$

Gdzie:

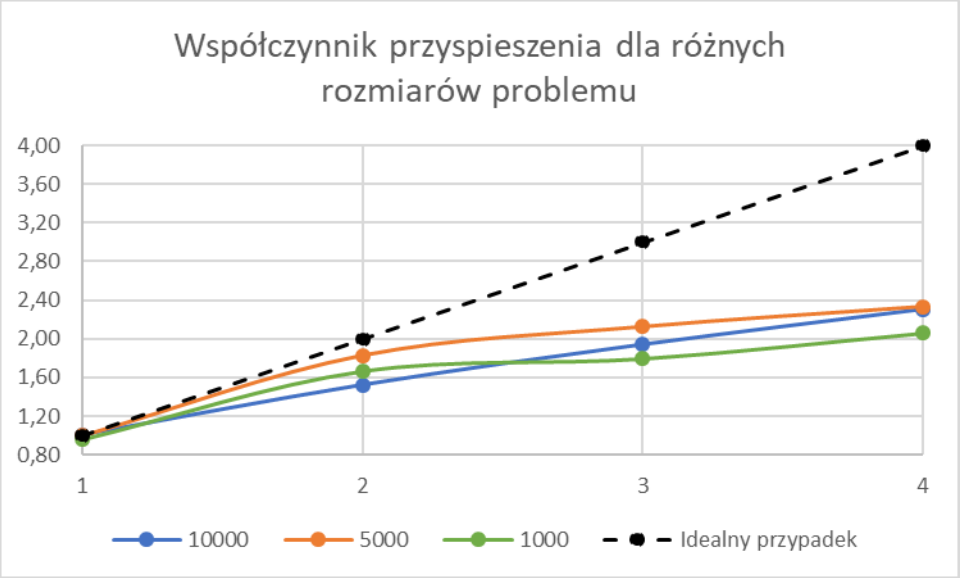
$S(n,p)$ – współczynnik przyspieszenia wykonania programu realizującego algorytm dla zadania o wielkości n na maszynie równoległej z p procesorami,

$T(n,1)$ - czas wykonania programu realizującego algorytm dla zadania o wielkości n na maszynie równoległej z 1 procesorem,

$T(n,p)$ – czas wykonania programu realizującego ten sam algorytm dla zadania o wielkości n na maszynie równoległej z p procesorami.

Tabela 2. Współczynnik przyspieszenia względem czasu wykonania części sekwncyjnej [-]

Rozmiar problemu [liczba wierszy macierzy]	*Część sekwencyjna*	*OpenMP nth=1*	*OpenMP nth=2*	*OpenMP nth=3*	*OpenMp nth=4*
10000	-	1,00	1,52	1,94	2,30
5000	-	1,00	1,83	2,13	2,33
1000	-	0,96	1,67	1,79	2,06



3. Wnioski

Uzyskano przyśpieszenie wykonywania operacji na poziomie 230% względem części sekwencyjnej (przy 4 wątkach). Należy pamiętać, że procesor na którym wykonywany jest program posiada dwa fizyczne rdzenie, jednakże wykorzystuje technologię Hyper-Threading, która tworzy dodatkowe procesory wirtualne, których skalowalność w obliczeniach równoległych nie jest tak wysoka, jak fizycznych rdzeni[2]. Stąd zmiana współczynnika przyśpieszenia między wykorzystaniem 2 rdzeni względem wykorzystania jednego jest największa.

Warto zwrócić uwagę, iż skompilowany program sekwencyjny wykonuje się szybciej, niż skompilowany program z wykorzystaniem dyrektyw OpenMP przy wywołaniu programu dla liczby wykorzystywanych wątków równej jeden. Jest to związane z dodatkowym narzutem czasowym, jakiego wymaga OpenMP po to, aby przeprowadzić proces zrównoleglenia. Dlatego też praktycznie niemożliwe jest uzyskanie współczynnika przyśpieszenia $P(n_{threads}) = n_{threads}$ (idealny przypadek na powyższym wykresie).

[2] https://en.wikipedia.org/wiki/Hyper-threading