МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Навчально-науковий Інститут

Комп’ютерних Інформаційних технологій

Кафедра комп’ютеризованих систем управління

Лабораторна робота№3

З дисципліни “Паралельні та розподілені обчислення”

На тему “ **Програмування для комп’ютерних систем зі спільною пам’яттю**”

Виконав:

Студент ННІКІТ СП-424

Єлістраток К.Є

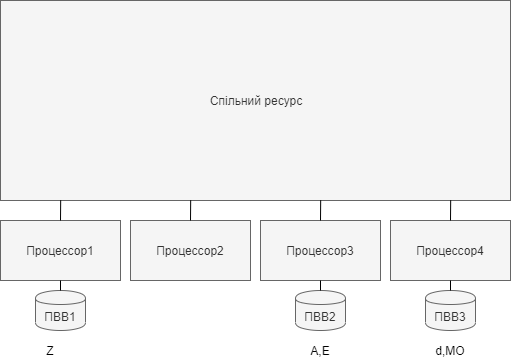
Перевірив:

Корочкін О В

Київ 2019

**ЗАВДАННЯ НА РОБОТУ**

**1. Структура паралельної комп’ютерної системи**



1. **Математична задача**

A = sort(d \* E + Z \* MO)

1. **Мова програмування:** C++
2. **Засоби програмування взаємодії потоків (задач)** : Секції open mp (#pragma omp section) **,** визначення номеру потоку

(omp\_get\_thread\_num)

**ВИКОНАННЯ РОБОТИ**

**Етап 1. Побудова паралельного алгоритму**

1 Uh = (d \* Eh + Z \* MOh)

2 Ah’ = Sort(Uh)

3 Ah = Sort(Ah’)

Спільні ресурси:  *d,*Z

**Етап 2. Розроблення алгоритмів потоків (задач) .**

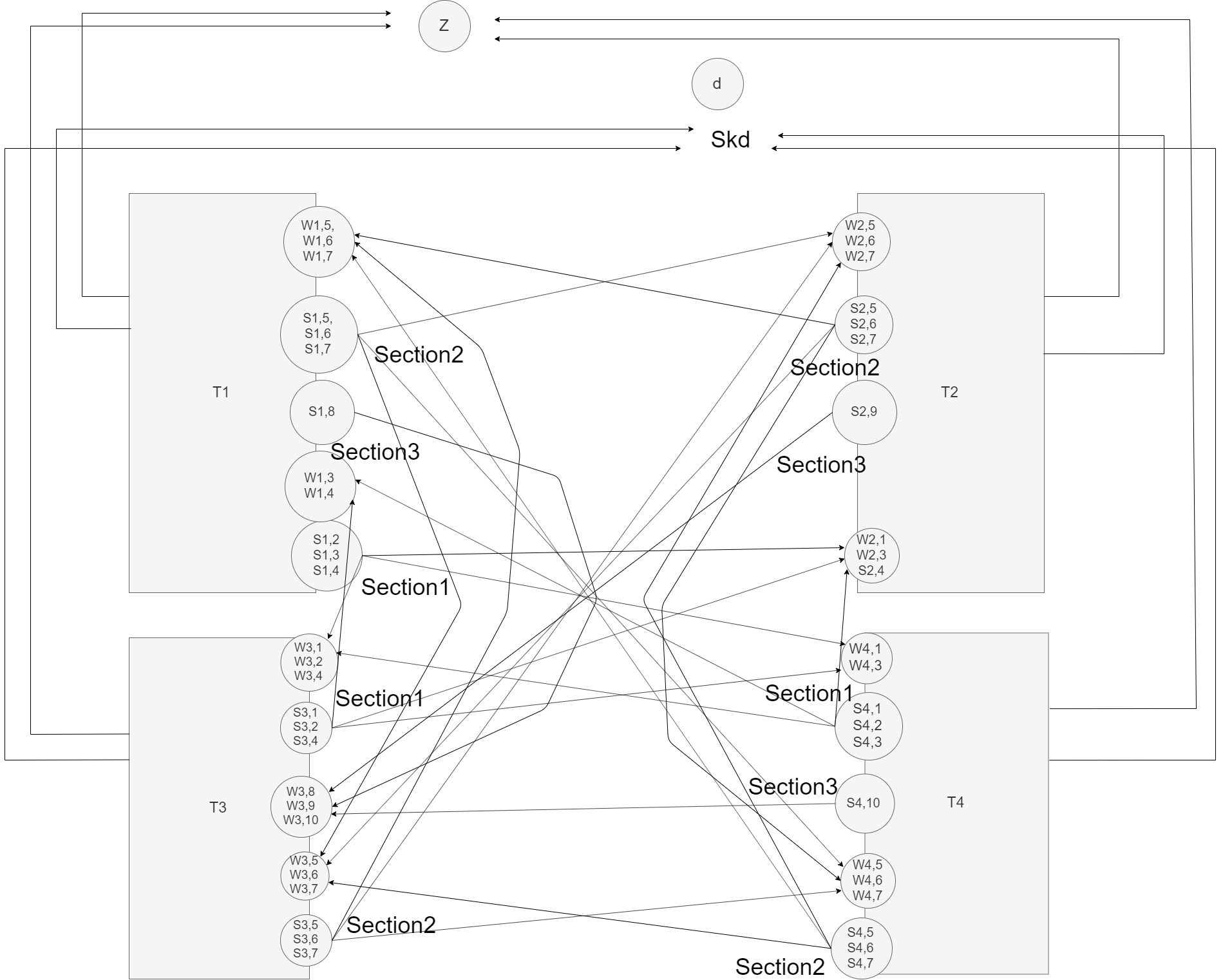
|  |  |  |
| --- | --- | --- |
| **№** | **Задача *Т1*** | ***Точки синхронізації*** |
| **1** | Введення Z |  |
| **2** | Сигнал T2,T3,T4 про ввід Z | S12,S13,S14 |
| **3** | Очікування сигналу T3,T4 про ввід | W13,W14 |
| **4** | d1 = d | КД |
| **5** | Uh = (d \* Eh + Z \* MOh) |  |
| **6** | Сигнал T2,T3,T4 про завершення розрахунку U | S15,S16,S17 |
| **7** | Очікування сигналу T2,T3,T4 про завершення розрахунку U | W15,W16,W17 |
| **8** | Ah’ = Sort(Uh) |  |
| **9** | Сигнал T3 про завершення розрахунку Ah’ | S18 |

|  |  |  |
| --- | --- | --- |
| **№** | **Задача *Т2*** | ***Точки синхронізації*** |
| **1** | Очікування сигналу T1,T3,T4 про ввід | W21,W23,W24 |
| **2** | d2 = d | КД |
| **3** | Uh = (d2 \* Eh + Z \* MOh) |  |
| **4** | Сигнал T2,T3,T4 про завершення розрахунку U | S25,S26,S27 |
| **5** | Очікування сигналу T2,T3,T4 про завершення розрахунку U | W25,W26,W27 |
| **6** | Ah’ = Sort(Uh) |  |
| **7** | Сигнал T3 про завершення розрахунку Ah’ | S2,9 |

|  |  |  |
| --- | --- | --- |
| **№** | **Задача *Т3*** | ***Точки синхронізації*** |
| **1** | Введення E |  |
| **2** | Сигнал T2,T1,T4 про ввід E | S32,S32,S34 |
| **3** | Очікування сигналу T1,T4 про ввід | W31,W34 |
| **4** | d3 = d | КД |
| **5** | Uh = (d3 \* Eh + Z \* MOh) |  |
| **6** | Сигнал T2,T1,T4 про завершення розрахунку U | S35,S36,S37 |
| **7** | Очікування сигналу T2,T1,T4 про завершення розрахунку U | W35,W36,W37 |
| **8** | Ah’ = Sort(Uh) |  |
| **9** | Очікування сигналу T1,T3,T4 про завершення розрахунків Ah’ | W3,8;W3,9;W3,10; |
| **10** | A = sort (Ah’) |  |
| **11** | Вивід A |  |

|  |  |  |
| --- | --- | --- |
| **№** | **Задача *Т4*** | ***Точки синхронізації*** |
| **1** | Введення d,MO |  |
| **2** | Сигнал T1,T2,T3 про ввід Z | S41,S42,S43 |
| **3** | Очікування сигналу T3,T1 про ввід | W43,W41 |
| **4** | d4 = d | КД |
| **5** | Uh = (d4\* Eh + Z \* MOh) |  |
| **6** | Сигнал T2,T3,T4 про завершення розрахунку U | S4,5;S4,6;S4,7 |
| **7** | Очікування сигналу T2,T3,T4 про завершення розрахунку U | W4,5;W4,6;W4,7 |
| **8** | Ah’ = Sort(Uh) |  |
| **9** | Сигнал T2 про завершення розрахунку Ah’ | S4,10 |

**Етап 3. Розроблення структурної схеми взаємодії задач.**



Призначення Секцій:

* ***Section1*** – для синхронізації по завершенню введення в *Т*1,T3,T4*;*
* ***Section2*** – для синхронізації по завершенню підрахунку U*;*
* ***Section3*** – для синхронізації по завершенню підрахунку Ah’*;*

Призначення Критичних Секцій :

***SKD*** – для керування доступом до спільного ресурсу (*d*);

*// lab3 c++ openmp*

*// Студента групи СП-424*

*// Єлістратова Кирила*

*// 24.11.19*

*#include "omp.h"*

*#include <iostream>*

*#include <algorithm>*

*#include <iterator>*

*#include <functional>*

*#include <vector>*

*const int N = 20, P = 4, h = N / P;*

*int Z[N], E[N],U[N];*

*int MO[N][N];*

*int d,d1,d2,d3,d4,dtmp;*

*std::vector< int > A;*

*// BEGIN OF INPUT DATA*

*void input\_data(int num) {*

*// THREAD 1 INPUT*

*if (num == 0) {*

*for (int i = 0; i < N; i++) {*

*Z[i] = 1;*

*}*

*}*

*// THREAD 3 INPUT*

*if (num == 2) {*

*int A[N][N];*

*for (int i = 0; i < N; i++) {*

*E[i] = 1;*

*}*

*}*

*// THREAD 4 INPUT*

*if (num == 3) {*

*d = 1;*

*for (int i = 0; i < N; i++) {*

*for (int j = 0; j < N; j++) {*

*MO[i][j] = 1;*

*}*

*}*

*}*

*std::cout<<num; //INPUT Num OF THREAD*

*}*

*// END OF INPUT DATA*

*void dublicate\_d(num){*

*if (num == 0) {*

*d1 = d;*

*}*

*if (num == 1) {*

*d2 = d;*

*}*

*if (num == 2) {*

*d3 = d;*

*}*

*if (num == 3) {*

*d4 = d;*

*}*

*}*

*// BEGIN OF CALCULATING U*

*void Calculate\_U(int start,int end) {*

*// THREAD CALCULATE*

*if (num == 0) {*

*tmpd = d1;*

*}*

*if (num == 1) {*

*tmpd = d2;*

*}*

*if (num == 2) {*

*tmpd = d3;*

*}*

*if (num == 3) {*

*tmpd = d4;*

*}*

*for (int i = start; i < end; i++)*

*{*

*int tmp = 0;*

*for (int j = 0; j < N; j++)*

*{*

*for (int k = 0; k < N; k++)*

*{*

*tmp += Z[k] \* MO[i][j];*

*}*

*}*

*U[i] = tmpd \* E[i] + tmp;*

*}*

*}*

*// END OF CALCULATE U*

*//Calculate A i*

*void Sort\_U(int start, int end) {*

*std::partial\_sort(start,end,h,U);*

*}*

*//End Calculate A i*

*// BEGIN OUTPUT*

*void OUTP(int num) {*

*if (num == 2) {*

*int tmp[N];*

*int g = 0;*

*tmp[0] = U[0];*

*tmp[1] = U[h - 1];*

*tmp[2] = U[h \* 2 - 1];*

*tmp[3] = U[h \* 3 - 1];*

*int tmp2 = 10000000;*

*for (int i = 0; i < P; i++) {*

*for (int j = 0; j < P; j++) {*

*if (tmp[j] < tmp2) {*

*g = j;*

*tmp2 = tmp[j];*

*}*

*for (int k = tmp2; k < tmp2 + h; k++) {*

*A.push\_back(U[i]);*

*delete [] tmp[g];*

*}*

*}*

*}*

*}*

*for (int i = 0; i < N; i++) {*

*std::cout << A[i];*

*}*

*}*

*// END OUTPUT*

*int main() {*

*int num;*

*#pragma omp parralel num\_threads(P)*

*//Begin Sections for Input*

*#pragma omp section*

*{*

*num = omp\_get\_thread\_num(); // GEt NUMBER OF THREAD*

*input\_data(num); // START INPUT FOR THIS THREAD*

*}*

*//End Sections for Input*

*//Begin Dublicating d*

*#pragma omp section*

*{*

*#pragma omp critical {*

*num = omp\_get\_thread\_num();*

*dublicate\_d(num);*

*}*

*}*

*//End Dublicating d*

*//Begin Sections for Calculating U*

*#pragma omp section*

*{*

*int start,end;*

*num = omp\_get\_thread\_num(); // GET NUMBER OF THREAD*

*if (num == 0) {*

*start = 0; end = h;*

*Calculate\_U(start, end);*

*}*

*if (num == 1) {*

*start = h; end = h\*2;*

*Calculate\_U(start, end);*

*}*

*if (num == 2) {*

*start = h\*2; end = h \* 3;*

*Calculate\_U(start, end);*

*}*

*if (num == 3) {*

*start = h \* 3; end = N;*

*Calculate\_U(start, end);*

*}*

*}*

*//End Sections for Calculating U*

*//Begin Section to calculate Ai*

*#pragma omp section*

*{*

*int start =0 , end =0;*

*num = omp\_get\_thread\_num(); // GET NUMBER OF THREAD*

*if (num == 0) {*

*start = 0; end = h;*

*Sort\_U(start, end);*

*}*

*if (num == 1) {*

*start = h; end = h \* 2;*

*Sort\_U(start, end);*

*}*

*if (num == 2) {*

*start = h \* 2; end = h \* 3;*

*Sort\_U(start, end);*

*}*

*if (num == 3) {*

*start = h \* 3; end = N;*

*Sort\_U(start, end);*

*}*

*}*

*//End Section to calculate Ai*

*//Begin of Sorting*

*#pragma omp section*

*{*

*num = omp\_get\_thread\_num(); // GET NUMBER OF THREAD*

*OUTP(num);*

*}*

*return 5;*

*}*