МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

**По лабораторной работе №3**  
по дисциплине «Операционные системы и системное программирование»

Тема: «СИНХРОНИЗАЦИЯ ПРОЦЕССОВ И ПОТОКОВ В ОС WINDOWS»

Выполнил: Казак И. Ю.

Группа: УИС-311

Преподаватель: доц. Варфоломеев В. А.

Борисенко Ф. А.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ВАРИАНТ ЗАДАНИЯ 2](#_Toc167645467)

[ТЕКСТ ПРОГРАММЫ 4](#_Toc167645468)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 15](#_Toc167645469)

[ВРЕМЕННЫЕ ДИАГРАММЫ ТРАСС ПОТОКОВ 16](#_Toc167645470)

[ВЫВОД 17](#_Toc167645471)

# ВАРИАНТ ЗАДАНИЯ

Таблица 1 – Варианты заданий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Разделяемые переменные, (тип) | Операция 1 | | Операция 2 | |
| Клавиши (N) | Действия | Клавиши (N) | Действия |
| 8 | R (str) = “ABCDE” | ← | R+1 | → | R-1 |

# ТЕКСТ ПРОГРАММЫ

#include <iostream>

#include <conio.h>

#include <windows.h>

#include <ctime>

#include <iomanip>

#include <mutex>

int R = 0;

clock\_t delayTime;

HANDLE hThread;

HANDLE mutex;

DWORD threadID;

struct LogJournal

{

    int operationNumber;

    int pressedKeyCode;

    DWORD ThreadID;

    clock\_t enteringCriticalSectionTime;

    clock\_t leavingCriticalSectionTime;

    clock\_t operationStartTime;

    clock\_t operationEndTime;

    clock\_t waitingTime;

    int R;

    int Rc;

} logJournal[1000];

void delay(clock\_t delayTime)

{

    clock\_t start = clock();

    while (clock() - start < delayTime)

        ;

}

bool isKeyPressed()

{

    return (GetAsyncKeyState(VK\_LEFT) & 0x8000) || (GetAsyncKeyState(VK\_RIGHT) & 0x8000);

}

int getKeyCode()

{

    int currentKey = 0;

    if (GetAsyncKeyState(VK\_LEFT) & 0x8000)

    {

        currentKey = VK\_LEFT;

        while (GetAsyncKeyState(VK\_LEFT) & 0x8000)

        {

            Sleep(50);

        }

    }

    else if (GetAsyncKeyState(VK\_RIGHT) & 0x8000)

    {

        currentKey = VK\_RIGHT;

        while (GetAsyncKeyState(VK\_RIGHT) & 0x8000)

        {

            Sleep(50);

        }

    }

    return currentKey;

}

DWORD WINAPI operationThreadProc(LPVOID param)

{

    WaitForSingleObject(mutex, INFINITE);

    LogJournal \*logEntry = static\_cast<LogJournal \*>(param);

    logEntry->enteringCriticalSectionTime = clock();

    bool isLeft = (logEntry->pressedKeyCode == VK\_LEFT);

    delay(delayTime);

    if (isLeft)

    {

        R += 1;

    }

    else

    {

        R -= 1;

    }

    delay(delayTime);

    logEntry->R = R;

    logEntry->operationEndTime = clock();

    logEntry->waitingTime = logEntry->operationEndTime - logEntry->operationStartTime;

    logEntry->leavingCriticalSectionTime = clock();

    ReleaseMutex(mutex);

    return 0;

}

void displayLogJournalHeader()

{

    std::cout << std::setw(5) << "No" << " | "

              << std::setw(4) << "Key" << " | "

              << std::setw(10) << "Thread ID" << " | "

              << std::setw(11) << "Start Time" << " | "

              << std::setw(9) << "End Time" << " | "

              << std::setw(10) << "Enter CS" << " | "

              << std::setw(10) << "Leave CS" << " | "

              << std::setw(10) << "Wait Time" << " | "

              << std::setw(7) << "R" << " | "

              << std::setw(7) << "Rc" << std::endl;

    std::cout << "--------------------------------------------------------------------------------------------------------------------" << std::endl;

}

void displayLogJournalEntry(const LogJournal &entry)

{

    std::cout << std::setw(5) << entry.operationNumber << " | "

              << std::setw(4) << entry.pressedKeyCode << " | "

              << std::setw(10) << entry.ThreadID << " | "

              << std::setw(11) << entry.operationStartTime << " | "

              << std::setw(9) << entry.operationEndTime << " | "

              << std::setw(10) << entry.enteringCriticalSectionTime << " | "

              << std::setw(10) << entry.leavingCriticalSectionTime << " | "

              << std::setw(10) << entry.waitingTime << " | "

              << std::setw(7) << entry.R << " | "

              << std::setw(7) << entry.Rc << std::endl;

}

int main(int argc, char \*argv[])

{

    SetConsoleCP(1251);

    SetConsoleOutputCP(1251);

    mutex = CreateMutex(NULL, false, NULL);

    if (argc != 2)

    {

        printf("Usage: %s <delay time in ms>\n", argv[0]);

        return 1;

    }

    delayTime = atoi(argv[1]);

    int operationIndex = 0;

    int displayIndex = 1;

    bool running = true;

    int controlR = R;

    logJournal[1].operationEndTime = 0;

    displayLogJournalHeader();

    while (running && operationIndex < 1000)

    {

        if (isKeyPressed())

        {

            int currentKeyCode = getKeyCode();

            operationIndex++;

            logJournal[operationIndex].operationNumber = operationIndex;

            logJournal[operationIndex].pressedKeyCode = currentKeyCode;

            logJournal[operationIndex].operationEndTime = 0;

            logJournal[operationIndex].R = 0;

            logJournal[operationIndex].operationStartTime = clock();

            hThread = CreateThread(NULL, 0, operationThreadProc, &logJournal[operationIndex], CREATE\_SUSPENDED, &threadID);

            bool isLeft = (currentKeyCode == VK\_LEFT);

            controlR = isLeft ? controlR + 1 : controlR - 1;

            logJournal[operationIndex].ThreadID = threadID;

            logJournal[operationIndex].Rc = controlR;

            ResumeThread(hThread);

        }

        if (logJournal[displayIndex].operationEndTime != 0)

        {

            displayLogJournalEntry(logJournal[displayIndex]);

            displayIndex++;

        }

    }

    CloseHandle(mutex);

    return 0;

}

# РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

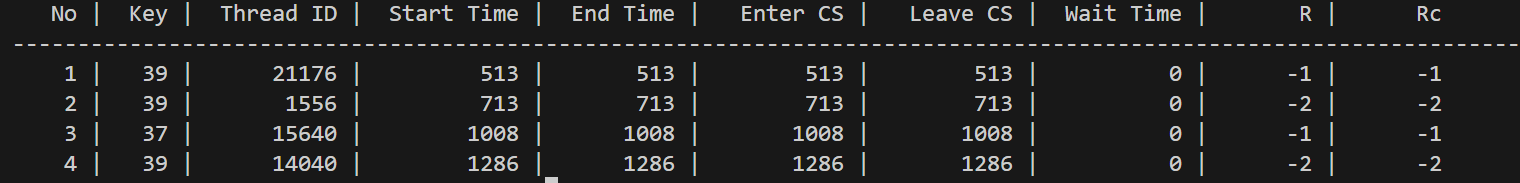


Рисунок 1 – Работа при задержке 0мс

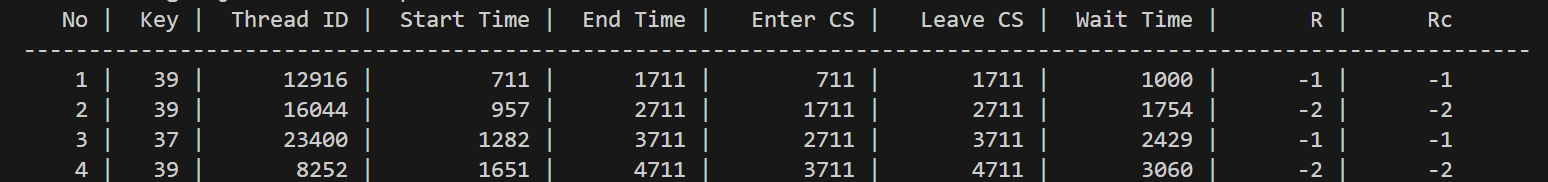
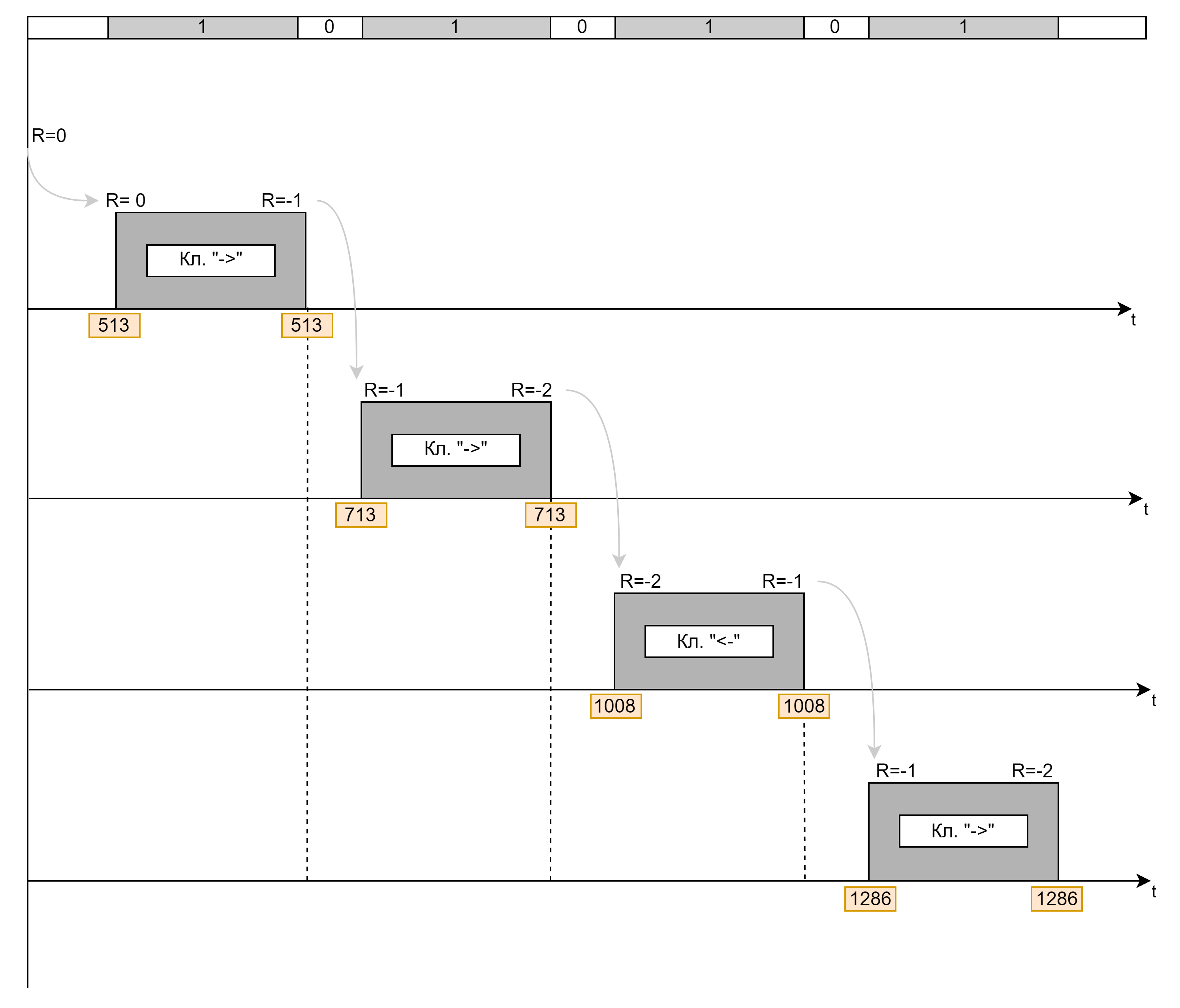
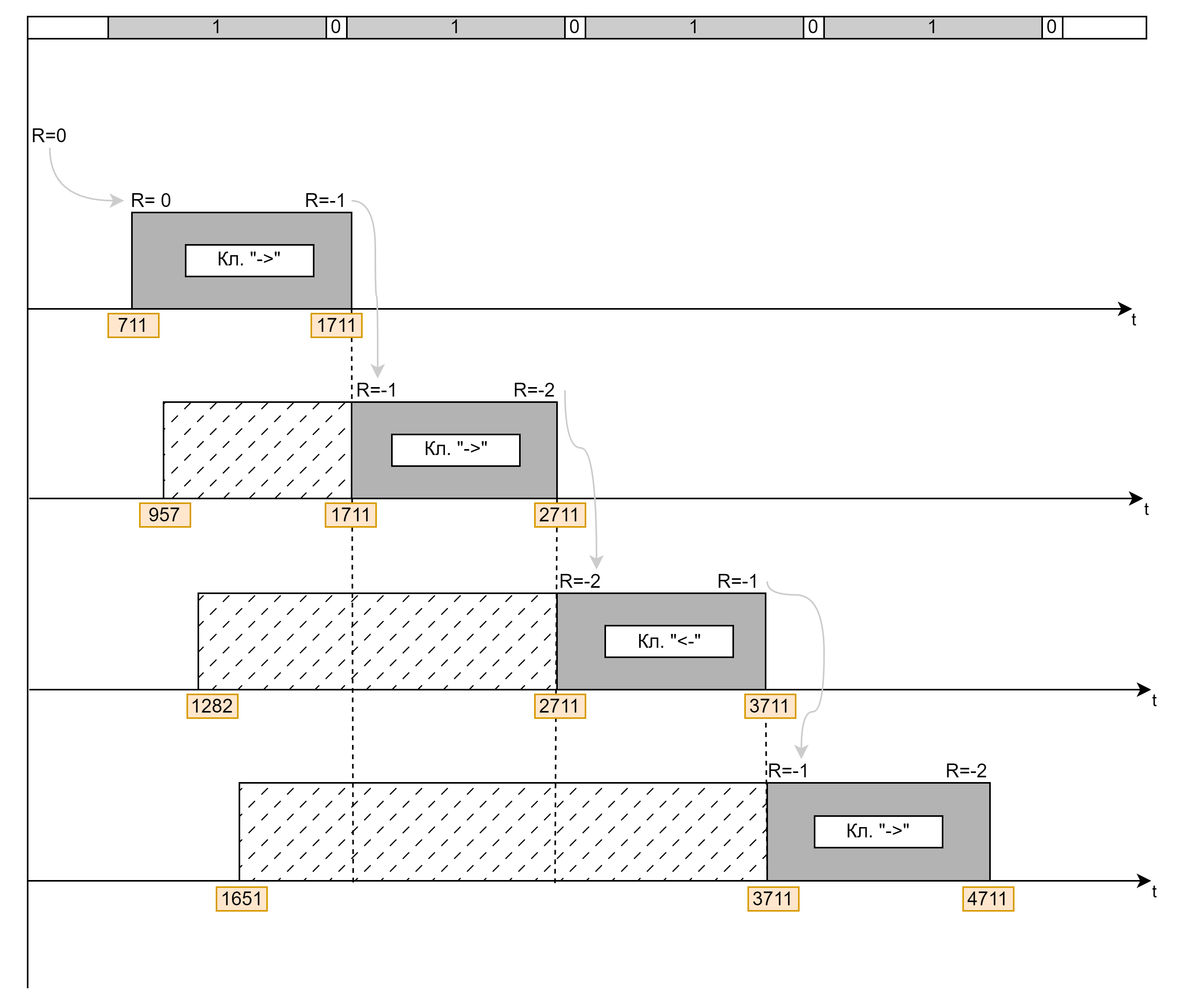


Рисунок 2 – Работа при задержке 500мс

# ВРЕМЕННЫЕ ДИАГРАММЫ ТРАСС ПОТОКОВ





# ВЫВОД

В ходе выполнения лабораторной работы была реализована многопоточная программа, в которой несколько потоков выполняют операции с глобальной переменной R. Для синхронизации доступа к этой переменной использовались мьютексы. Это позволило избежать ситуаций, когда несколько потоков одновременно пытаются изменить значение переменной, что могло бы привести к некорректным результатам и непредсказуемому поведению программы.

Одной из ключевых задач было обеспечение защиты критической секции, в которой происходят изменения глобальной переменной R. Это было достигнуто с помощью мьютекса, который блокировал доступ к критической секции для других потоков, пока один из них выполняет в ней операции. Таким образом, мьютексы обеспечили правильную последовательность выполнения потоков и предотвратили гонки данных.

В программе был реализован журнал логирования, который записывал информацию о каждой выполненной операции. В логах фиксировались номера операций, коды нажатых клавиш, идентификаторы потоков, времена начала и окончания операций, а также времена входа и выхода из критической секции. Это позволило подробно отслеживать выполнение программы и анализировать работу потоков.

На основе логов можно сделать вывод, что использование мьютексов значительно улучшает безопасность многопоточных программ. В данном случае, благодаря синхронизации, удалось избежать некорректных состояний переменной R, что подтверждает эффективность выбранного метода.